,VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI - 590018

A

Project Report

on

PriceProbe: E-Commerce Platforms Using Machine Learning

[18AIP77]

Submitted in partial fulfillment for the award of degree

BACHELOR OF ENGINEERING

in

Department of Artificial Intelligence and Machine Learning

by

A S Sushmitha Urs	**1JT20AI001**
Abhishek Kumar Pandey	**1JT20AI002**
Jagruthi G	**1JT20AI013**
Vaibhavi B Raj	**1JT20AI047**

Under the Guidance of

Prof. Archana VR

Assistant Professor

Department of AI&ML

Department of Artificial Intelligence and Machine Learning

Jyothy Institute of Technology

Tataguni, Off Kanakapura Road, Bangalore-560 082

Academic Year 2023-2024

JYOTHY INSTITUTE OF ENGINEERING AND TECHNOLOGY BENGALURU-560082, KARNATAKA

Department of Artificial Intelligence and Machine Learning

CERTIFICATE

Certified that the project work entitled ***"PriceProbe: E-Commerce Platforms Using Machine Learning "*** carried out by **A S Sushmitha Urs [1JT20AI001], Abhishek Kumar Pandey [1JT20AI002], Jagruthi G [1JT20AI013], Vaibhavi B Raj [1JT20AI047],**a bonafide student of **Jyothy Institute of Technology, Bangalore** in partial fulfillment for the award of Bachelor of Engineering / Bachelor of Technology in **Artificial Intelligence and Machine Learning** of the Visveswaraiah Technological University, Belgaum during the year **2023-2024.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

Prof. Archana VR Assistant Professor Dept. of AI&ML JIT	**Dr. Madhu B R** Professor and Head Dept. of AI&ML, JIT	**Dr. K. Gopalakrishna** Principal JIT

PRINCIPAL
Jyothy Institute of Technology
Thathaguni, Off Kanakapura Main Road
Bengaluru - 560 082

EXTERNAL VIVA

Name of the examiners	Signature with date
1. Mr. Mahesh Basavaraj	Mahesh B 28/5
2. Mrs. Soumya KN	Soumya KN 28/5

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to Jyothy Institute of Technology, VTU for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

In particular we would like to thank ***Dr. K Gopalkrishna****, Principal, Jyothy Institute of Technology, VTU for their constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr. Madhu B R, Professor and Head of the department, Artificial Intelligence and Machine Learning,*** *for providing right academic guidance that made our task possible.*

We would like to thank our guide ***Prof. Archana VR*** *Assistant Professor,* ***Dept. of Artificial Intelligence and Machine Learning****, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

We would like to thank our Project Coordinator ***Prof. Soumya K N*** *and all the staff members AIML for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Signature of Students

Sushmitha

Abhishek

Jagruthi

Raj

DECLARATION

We, **A S Sushmitha Urs [1JT20AI001], Abhishek Kumar Pandey [1JT20AI002], Jagruthi G [1JT20AI013], Vaibhavi B Raj [1JT20AI047]** are students of Eight semester B.Tech in **Artificial Intelligence and Machine Learning** at Jyothy Institute of Technology, **VTU**, hereby declare that the project titled "***PriceProbe: E-Commerce Platforms Using Machine Learning***" has been carried out by us and submitted in partial fulfilment for the award of degree in Bachelor of Technology in **Artificial Intelligence and Machine Learning** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Signature

A S Sushmitha Urs:
1JT20AI001

Abhishek Kumar Pandey:
1JT20AI002

Jagruthi G:
1JT20AI013

Vaibhavi B Raj:
1JT20AI047

Place: Bangalore
Date :

ABSTRACT

This research is an innovative chatbot revolutionizing the e-commerce landscape. By harnessing advanced algorithms and real-time data analysis, it simplifies online shopping by providing users with comprehensive comparisons and exclusive discounts from various platforms. Its adaptive nature enables personalized recommendations, learning from user interactions to enhance relevance and responsiveness. This project empowers users to make informed decisions, maximizing convenience and savings in the dynamic world of online retail.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1INTRODUCTION:

In the realm of e-commerce, where the search for the best deals can feel like navigating a maze, the PriceProbe Chatbot emerges as a beacon of simplification. This revolutionary tool leverages advanced algorithms and real-time data analysis to streamline the shopping experience. By seamlessly integrating with various platforms, it rapidly sifts through vast amounts of data to offer users timely comparisons and access to exclusive discounts.

Central to the PriceProbe Chatbot's efficacy is its adaptive nature, distinguishing it as a dynamic ally for shoppers. Through continuous learning from user interactions, the chatbot refines its recommendations to remain relevant and responsive to individual preferences. Strategies honed through the analysis of customer behavior ensure that the chatbot effectively attracts engagement and enhances user satisfaction.

Amidst the multitude of e-commerce options available, PriceProbe Chatbot stands out by empowering users with tailored insights. By guiding them towards informed decisions, it alleviates the challenge of choosing the best deal across various websites. This personalized approach not only saves users valuable time but also enables them to capitalize on cost-effective opportunities that might otherwise be overlooked.

In a landscape where convenience and savings reign supreme, PriceProbe Chatbot becomes an indispensable asset. By consolidating information from disparate sources and distilling it into actionable insights, the chatbot eliminates the need for exhaustive manual searches. This efficiency not only streamlines the shopping process but also ensures that users make the most of available discounts and offers.

Whether users are in pursuit of the latest gadgets or everyday essentials, PriceProbe Chatbot serves as a trusted companion in the quest for value. Its tailored recommendations and access to exclusive

discounts facilitate a seamless shopping experience marked by convenience and savings. With its ability to adapt and learn from user interactions, the chatbot continuously enhances its value as an indispensable tool for online shoppers.

In addition to its role in simplifying the shopping experience, PriceProbe Chatbot also serves as a catalyst for innovation within the e-commerce landscape. By continuously analyzing user interactions and market trends, it not only enhances its own capabilities but also provides valuable insights to retailers and platform providers. This symbiotic relationship fosters a cycle of improvement, driving the evolution of e-commerce platforms to better meet the needs and preferences of consumers. As PriceProbe Chatbot continues to revolutionize the way users shop online, it simultaneously fuels advancements in the industry, ultimately shaping a more efficient and user-centric e-commerce ecosystem.

CHAPTER 2
LITERATURE SURVEY

1. An E-Commerce Control Unit for Addressing Online Transactions in Developing Countries: Saudi Arabia—Case Study Omar Saeed Al-Mushayt; Wajeb Gharibi; Nasrullah Armi

The Online transactions play an increasingly important role in our daily lives. Recently, onlineshopping has dramatically expanded not only in small and medium enterprises, but also among individual internet users who use social media as online trading platforms. While there are several online-shopping platforms in Saudi Arabia, they are still facing critical obstacles that challenge customers, businessmen, and organizations. This paper presents a smart control unit that could help address current challengesfacing e-commerce and suggest recent government legislation dedicated to governing and simplifying online transactions to make them more reliable, faster, secure, and competitive.

2. Product Comparison Website using Web scraping and Machine learning. Aswad Shaikh, Aniket Sonmali, Soham Wakade

The research paper details the creation of a product comparison website utilizing web scraping techniques to gather and analyze data from multiple product websites. By employing a customized algorithm, the website offers users comprehensive comparisons based on factors like price, features, and user ratings, aiding informed purchasing decisions. Furthermore, the developed platform serves as a prototype for similar websites across various categories.

3. E-commerce network with price comparator sites Ladislav Beranek; Radim Remes

The paper investigates e-commerce dynamics through bipartite graph modeling, focusing on customer-e-shop relationships and the role of price comparison sites like Heureka. It addresses market entry strategies, discontinuation timing for online pricing comparison services, and related issues using network analysis and simulation methods. Keywords include network-based inference, simulation, price comparison site, and e-commerce.

4. E-commerce Price Comparison Website Using Web Scraping Arman Shaikh , Raihan Khan , Komal Panokher , Mritunjay Kr Ranjan , Vaibhav Sonaje

This paper focuses on a Price Comparison website utilizing web scraping techniques, catering to consumers seeking cost-effective purchases amid busy lifestyles. By aggregating price data from various providers, the platform aims to streamline the shopping process, allowing users to make

informed decisions and save time and money. Keywords include Web Scraper, E-commerce, and Price Comparison.

5. Best Price - Product Comparison Android App For Online And Offline Market Prof. H.P. Bhabad, Atharva Vyavahare, Abhishek Dengale, Yogesh Hiwale, Mehul Gosavi

This abstract highlights top-rated price comparison apps, including Honey for discounts and PriceGrabber for comprehensive comparisons. CamelCamelCamel and Google Shopping cater to Amazon shoppers and offer broader searches across various retailers. Price Runner covers multiple categories with user reviews, enhancing convenience and cost savings for consumers seeking the best rates.

6. Scraping and Visualization of Product Data from E-commerce Websites V. Srividhya P.Megala

The paper explores "Scraping and Visualization of Product Data from E-commerce Websites," focusing on web scraping's cost-effectiveness, ease of implementation, and speed in extracting unstructured data for analysis. It comprises three phases: web scraping to store data in CSV format, data analysis using statistical methods, and visualization of extracted data through various charts, enhancing understanding and insights. Keywords include web scraping, data analysis, visualization, and data mining.

7. Price Comparison Website Prof. Harishchandra Maurya , Komal Patil , Shreya Sawant , Mrudula Thange , Asmita Mahadik

This Mobile apps have evolved to be more useful for regular use in recent years. The goal of this project is to give users an easy way to compare product availability and costs on various e-commerce websites. Users can easily compare prices from numerous sources by simplyentering the product information into the programme. To compare the product information found on several websites side by side, the application's databases are then searched. In order to ensure that they never miss out on a great offer, customers can also receive push notificationswhen things become available or go on sale.

8. Price Comparison for Products in Various ECommerce Website Mrs. M. Sowmiya, Srinandhan cs, Mugesh raja m, Sudheekshan kumar s

The abstract emphasizes the importance of price comparison websites in facilitating informed purchases across numerous e-commerce platforms. Employing web crawling and scraping techniques, users can efficiently compare prices and product features, enabling informed decision-making. The proposed system prioritizes user expectations and security, offering a simple and effective solution for accessing competitive prices and discount offers from various e-commerce websites.

CHAPTER 3
OBJECTIVE AND METHODOLOGY

3.1 OBJECTIVE

- Analysis and data preprocessing using webscraping.
- Design a scalable and modular system architecture that supports data ingestion, processing, analysis, and presentation layers.
- Evaluate the Price Probe. This tool helps users find discounts and save money by automatically comparing prices from several sellers. It also analyses user interests and behaviour to provide personalised recommendations.

3.2 METHODOLOGY

1. **Data preparation**

Initializing the essential libraries, such as requests and BeautifulSoup. Additionally, we define the headers variable, which incorporates a user agent. This user agent aids in emulating a web browser while sending requests to the websites.

2. **User Interaction**

The chatbot interface prompts users to input the desired product name, initiating the search process for personalized recommendations and tailored assistance.

3. **Streamlit**

It is a Python library designed for building web applications tailored to data science and machine learning projects. It simplifies the creation of interactive and customizable user interfaces, facilitating data visualization, analysis, and prediction tasks without extensive coding.With seamless integration with prominent data science libraries like Pandas and TensorFlow, Streamlit offers a user-friendly API for rapid development.

4. **Price Retrieval**

Implementing three functions, flipkart(), amazon(), and croma(), to search for the product on each website and retrieve the price. Each function uses the requests library to send a GET request to the website's search API and then parses the HTML response using BeautifulSoup.

5. Personalized Recommendation

Following price retrieval, the system incorporates personalized recommendation techniques to enhance user experience. This functionality tailors product suggestions based on user preferences and past interactions, optimizing shopping outcomes.

6. NLP Integration

Alongside personalized recommendations, Natural Language Processing (NLP) techniques are integrated to further refine user interactions. NLP facilitates the interpretation of user inputs, enabling the system to better understand and respond to user queries with enhanced accuracy and relevance.

7. Price Comparison

The system initiates a comparison process to identify the minimum price among the obtained data. Filtering out non-positive pricesensures accuracy, followed by calculating the minimum price for accurate cost assessment. This step ensures that users are presented with the most competitive pricing options available, optimizing their shopping experience.

8. URL Display

Printing the minimum price and the corresponding URL for the product on the website offering the lowest price. Also printing the URLs for the product on all three websites for reference.

CHAPTER 4
SYSTEM DESIGN

4.1 System Architecture

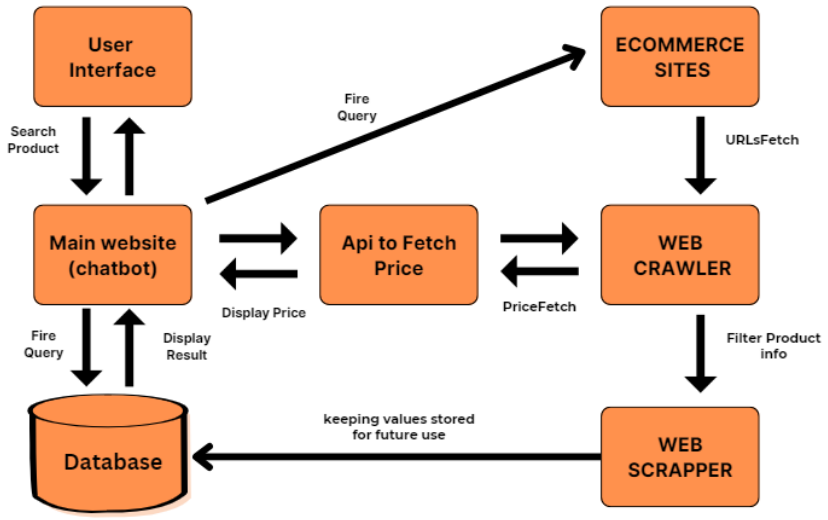


Fig 4.1.1 System architecture

The Fig.4.1.1 describes the search functionality, product information display, and price fetching processes of an ecommerce site. It involves various components such as user interface, main website, chatbot, database, web crawler, and web scraper. The processes include fire queries, URL fetching, filtering product info, and storing values for future use.

1. **Initiating a search or fire query:** The process begins when a user searches for a product, either by interacting with a user interface on a main website
   or through a conversational chatbot.

2. **Displaying the search result:** Based on the user's query, the system retrieves a list of relevant products and displays the search result to the user.

3. **Connecting to a database:** The search result is connected to a database that contains comprehensive product information, including product descriptions, specifications, and corresponding prices.

4. **Fetching the price:** To fetch the price of a product, the system uses an API (Application Programming Interface) to communicate with external services, which provide real-time price information.

5. **Displaying the price:** The fetched price is then displayed as part of the product information for the user to view.

6. **Storing values for future use:** The system may temporarily store data in a cache for fast access, which helps to improve the overall user experience by reducing the time needed to fetch and display information.

7. **Collecting product information:** The system uses a web crawler and web scraper to gather URLs and extract relevant product information from various e-commerce sites. This is an essential step in the data collection and pre-processing phase, ensuring that the system has up-to-date and accurate product information.

8. **Filtering product information:** After fetching the raw data, the system filters and processes the information to remove irrelevant data, correct errors, and format the information in a user-friendly manner.

4.2 Use Case Diagram

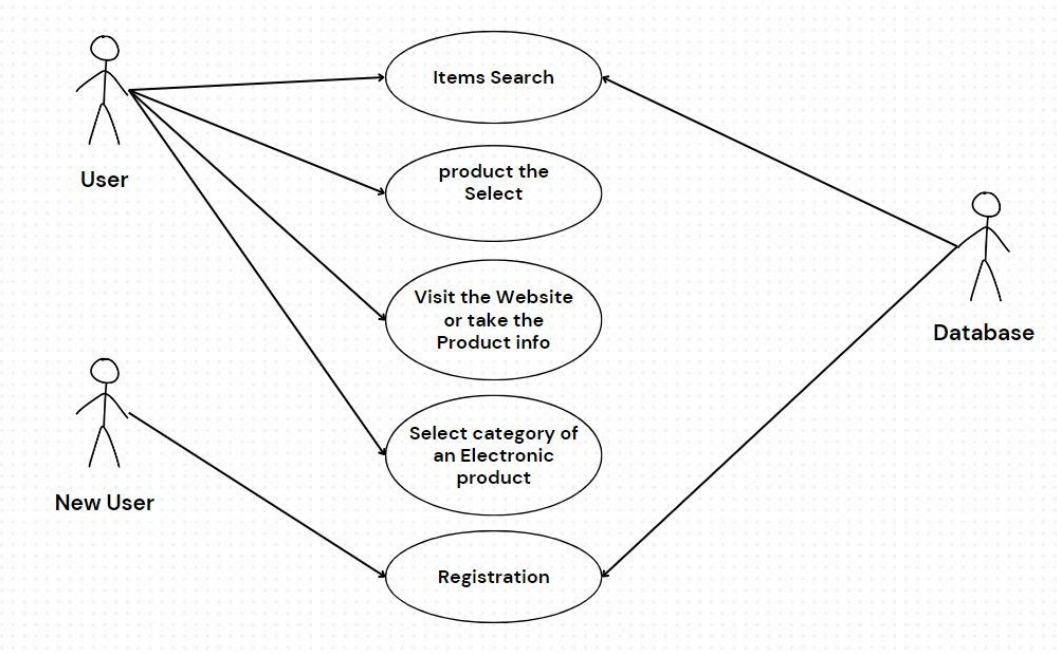


Fig 4.3.1 Use Case Diagram

The Fig.4.3.1 depicts the functionalities of an e-commerce website from the perspective of a customer. It includes use cases such as registering for an account, logging in, searching for products, adding products to the cart, making payments, and receiving email confirmations for orders and deliveries. The diagram highlights the interactions between the customer and the system, and the various functionalities available to the customer.

4.3 Data Flow Diagram

Fig 4.3.1 Data Flow chart

The Fig.4.3.1 illustrates the process of purchasing a product on an e-commerce website, from registering and logging in, to searching for a product, making a payment, and receiving email confirmations for order and delivery. The chart shows how data flows through the system, from customer input to system processing and communication back to the customer.

The functionalities of an e-commerce website from the perspective of a customer. The customer begins by registering for an account on the website, providing their personal information such as name, email, and password. Once registered, the customer can log into their account using their email and password. The customer can then search for products they want to purchase by entering keywords or browsing through categories. The system retrieves information about the product such as its name, price, description, and images, and displays it to the customer. The customer can add the product to their cart and proceed to checkout, where they enter their payment information. The system processes the payment and confirms the purchase, sending an email confirmation to the

customer. The system then arranges for the product to be delivered to the customer and sends an email confirmation for delivery. Finally, the customer can log out of their account.

4.4 Sequence Diagram

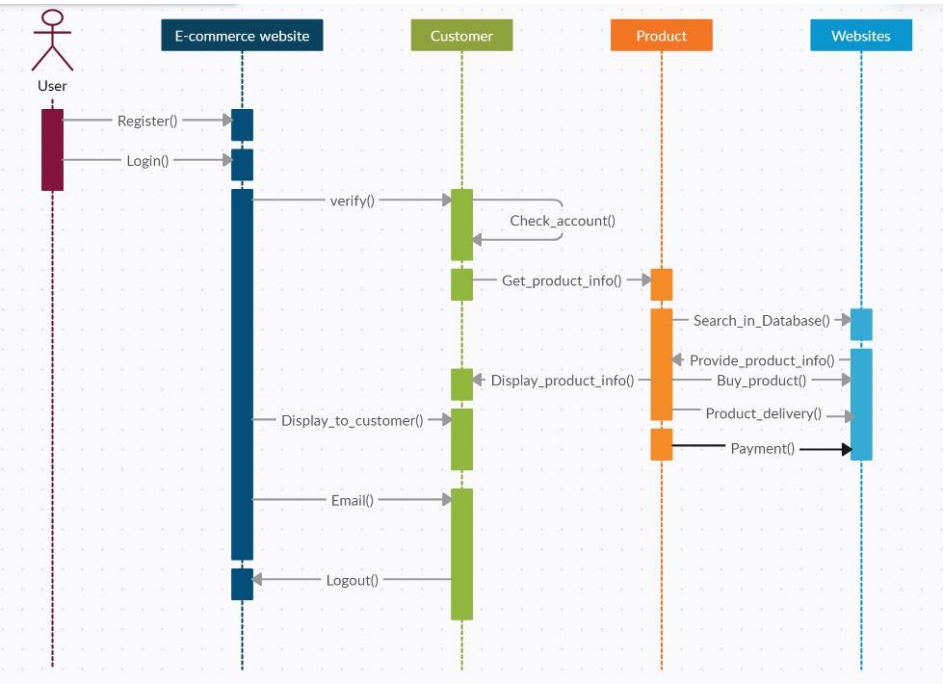


Fig 4.4.1 Sequence Flow diagram

The Fig.4.4.1 shows the process of a customer purchasing a product on an e-commerce website, from registering and logging in, to searching for a product, making a payment, and receiving an email confirmation. The system verifies the customer's account, retrieves product information, and arranges for delivery, while the customer provides payment information and confirms the purchase. The system also sends an email confirmation to the customer.

CHAPTER 5
IMPLEMENTATION

IMPLEMENTATION

Main:

```
import streamlit as st
from streamlit_option_menu import option_menu
import Account, About, Service, Contact, Chatbot, Home
st.set_page_config(
    page_title="PriceProbe", layout="wide", page_icon=":heavy_dollar_sign:"
)
class MultiApp:
    def __init__(self):
        self.apps = []
    def add_app(self, title, func):
        self.apps.append({
            "title": title,
            "function": func
        })
    def run(self):
        # app = st.sidebar(
        with st.sidebar:
            app = option_menu(
                menu_title='PriceProbe',
                options=['Home', 'Account', 'About', 'Service', 'Contact', 'ChatBot'],
                icons=['house-fill', 'person-circle', 'info-circle-fill', 'gear-wide-connected',
'telephone-fill', 'chat-left-dots-fill'],
                menu_icon='bag-heart',
                default_index=1,
                styles={
                    "container": {"padding": "5!important", "background-color":
'#403B83'},
                    "icon": {"color": "white", "font-size": "23px"},
```

```
                        "nav-link": {"color": "white", "font-size": "20px", "text-align": "left",
                                "margin": "0px", "--hover-color": "#36316D"},
                        "nav-link-selected": {"background-color": "#797AC5"}
                    }
                )
            if app == "Home":
                Home.app()
            if app == "Account":
                Account.app()
            if app == "About":
                About.app()
            if app == 'Service':
                Service.app()
            if app == 'Contact':
                Contact.app()
            if app == 'ChatBot':
                Chatbot.app()

multi_app = MultiApp()
# Add Chatbot as an app
multi_app.add_app('ChatBot', Chatbot.app)
# Run the app
multi_app.run()
custom_css = """
<style>
[data-testid="stSidebar"] > div:first-child {
   background-color: #797AC5; /* Sidebar background color */
}
.sidebar-image {
   width: 40px; /* Adjust width as needed */
   height: 0.2px; /* Maintain aspect ratio */
```

```
}
</style>
"""
st.markdown(custom_css, unsafe_allow_html=True)
# Assuming your image is named 'sidebar_image.png' and located in an 'images' folder within
your project
image_path = 'Images/Slide.png'
st.sidebar.image(image_path, use_column_width=True)
```

About:

```
import streamlit as st
import requests
from streamlit_lottie import st_lottie
from PIL import Image

def load_lottieurl(url):
    r = requests.get(url)
    if r.status_code != 200:
        return None
    return r.json()
lottie_chat = load_lottieurl("https://lottie.host/30a359ff-2a3d-4246-bff0-
650fa809897e/YvDHSGMDhj.json")
goal_img = Image.open("Images/about img.png")
def app():
    with st.container():
        left_column, right_column = st.columns(2)
        with left_column:
            st.header("Introduction to the Team")
            st.write("##")
            st.write(
            """
```

Our team at priceProbe is a dedicated group of individuals with a passion for technology and e-commerce.

From software engineers to data scientists, each member brings a unique skillset that drives our mission to provide top-notch service to our customers.

Together, we work tirelessly to enhance the user experience and deliver exceptional results.

""")

```
    with right_column:
        st_lottie(lottie_chat, height=300, key="coding")
with st.container():
    st.write(" --- ")
    image_column, text_column = st.columns((1, 2))
    with image_column:
        st.write(goal_img)
    with text_column:
        st.header("Purpose and Goals")
        st.write("##")
        st.write("""Our primary goal at priceProbe is to offer our customers a seamless shopping experience by comparing prices from various websites and providing personalized recommendations based on their preferences.We strive to help our customers save time and money by ensuring they receive the best deals available in the market. """)
with st.container():
    st.write(" --- ")
    text_column, image_column = st.columns((2, 1))
    with text_column:
        st.header("Offerings")
        st.write("##")
        st.write("""
```

At priceProbe, we offer a wide range of products across various categories, including electronics, fashion, home goods, and more.

Our platform uses machine learning algorithms to analyze pricing data and make personalized recommendations to help you find the best deals.

With our user-friendly interface, shopping has never been easier.

""")

with image_column:

st.write(goal_img)

st.image('Images/Websites.png', caption='Websites Available in our website', use_column_width=True)

Remember to call the app function to display the app

if __name__ == "__main__":

app()

Account:

```
import streamlit as st

import firebase_admin

from firebase_admin import firestore

from firebase_admin import credentials

from firebase_admin import auth

import json

import requests

cred = credentials.Certificate("priceprobe-2024-61ee385edea2.json")

firebase_admin.initialize_app(cred)

def app():

# Usernm = []

    st.title('Welcome to :violet[PriceProbe] :sunglasses:')

    if 'username' not in st.session_state:
```

```
        st.session_state.username = ''
    if 'useremail' not in st.session_state:
        st.session_state.useremail = ''
    def sign_up_with_email_and_password(email, password, username=None,
return_secure_token=True):
        try:
            rest_api_url = "https://identitytoolkit.googleapis.com/v1/accounts:signUp"
            payload = {
                "email": email,
                "password": password,
                "returnSecureToken": return_secure_token
            }
            if username:
                payload["displayName"] = username
            payload = json.dumps(payload)
            r = requests.post(rest_api_url, params={"key": "AIzaSyApr-
etDzcGcsVcmaw7R7rPxx3A09as7uw"}, data=payload)
            try:
                return r.json()['email']
            except:
                st.warning(r.json())
        except Exception as e:
            st.warning(f'Signup failed: {e}')
```

```
    def sign_in_with_email_and_password(email=None, password=None,
return_secure_token=True):
        rest_api_url = "https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword"
        try:
            payload = {
                "returnSecureToken": return_secure_token
            }
            if email:
                payload["email"] = email
            if password:
                payload["password"] = password
            payload = json.dumps(payload)
            print('payload sigin',payload)
            r = requests.post(rest_api_url, params={"key": "AIzaSyApr-
etDzcGcsVcmaw7R7rPxx3A09as7uw"}, data=payload)
            try:
                data = r.json()
                user_info = {
                    'email': data['email'],
                    'username': data.get('displayName')  # Retrieve username if available
                }
                return user_info
            except:
```

```
            st.warning(data)
        except Exception as e:
            st.warning(f'Signin failed: {e}')
    def reset_password(email):
        try:
            rest_api_url = "https://identitytoolkit.googleapis.com/v1/accounts:sendOobCode"
            payload = {
                "email": email,
                "requestType": "PASSWORD_RESET"
            }
            payload = json.dumps(payload)
            r = requests.post(rest_api_url, params={"key": "AIzaSyApr-
etDzcGcsVcmaw7R7rPxx3A09as7uw"}, data=payload)
            if r.status_code == 200:
                return True, "Reset email Sent"
            else:
                # Handle error response
                error_message = r.json().get('error', {}).get('message')
                return False, error_message
        except Exception as e:
            return False, str(e)
    # Example usage
    # email = "example@example.com"
```

```
    def f():
        try:
            # user = auth.get_user_by_email(email)
            # print(user.uid)
            # st.session_state.username = user.uid
            # st.session_state.useremail = user.email
            userinfo =
sign_in_with_email_and_password(st.session_state.email_input,st.session_state.password_input)
            st.session_state.username = userinfo['username']
            st.session_state.useremail = userinfo['email']
            global Usernm
            Usernm=(userinfo['username'])
            st.session_state.signedout = True
            st.session_state.signout = True
        except:
            st.warning('Login Failed')
    def t():
        st.session_state.signout = False
        st.session_state.signedout = False
        st.session_state.username = ''
    def forget():
        email = st.text_input('Email')
        if st.button('Send Reset Link'):
```

```
            print(email)
            success, message = reset_password(email)
            if success:
                st.success("Password reset email sent successfully.")
            else:
                st.warning(f"Password reset failed: {message}")
    if "signedout"  not in st.session_state:
        st.session_state["signedout"] = False
    if 'signout' not in st.session_state:
        st.session_state['signout'] = False
    if  not st.session_state["signedout"]: # only show if the state is False, hence the button has never been clicked
        choice = st.selectbox('Login/Signup',['Login','Sign up'])
        email = st.text_input('Email Address')
        password = st.text_input('Password',type='password')
        st.session_state.email_input = email
        st.session_state.password_input = password
        if choice == 'Sign up':
            username = st.text_input("Enter  your unique username")
            if st.button('Create my account'):
                # user = auth.create_user(email = email, password = password,uid=username)
                user = sign_up_with_email_and_password(email=email,password=password,username=username)
```

```
            st.success('Account created successfully!')
            st.markdown('Please Login using your email and password')
            st.balloons()
        else:
            # st.button('Login', on_click=f)
            st.button('Login', on_click=f)
            # if st.button('Forget'):
            forget()
            # st.button('Forget',on_click=forget)
    if st.session_state.signout:
            st.text('Name:'+st.session_state.username)
            st.text('Email id: '+st.session_state.useremail)
            st.button('Sign out', on_click=t)
    def ap():
        st.write('Posts')
```

ChatBot:

```
import streamlit as st
import base64
from comparison import compare_prices
def app(user_input=None):
    @st.cache_data
    def get_img_as_base64(file):
```

```
    with open(file, "rb") as f:
        data = f.read()
    return base64.b64encode(data).decode()
img = get_img_as_base64("Images/chatbot bg.png")
page_bg_img = f"""
<style>
[data-testid="stAppViewContainer"]::before {{
content: "";
position: absolute;
top: 0;
left: 0;
right: 0;
bottom: 0;
background-image: url("data:Images/chatbot bg.png;base64,{img}");
background-size: cover;
opacity: 100%;  # Adjust the opacity as needed
z-index: -1;  # Ensure it's in the background
}}
[data-testid="stHeader"] {{
background: rgba(0,0,0,0);
}}
</style>
"""
```

```
    st.markdown(page_bg_img, unsafe_allow_html=True)
    # Welcome message
    st.title("Welcome to the E-commerce Price Comparison Chatbot!")
    while True:
        user_input = st.text_input("You:", key="user_input_chatbot_1")
        if any(word in user_input.lower() for word in ["bye", "see you later", "goodbye"]):
            st.write("Thank you for using the E-commerce Price Comparison Chatbot. Goodbye!")
            break
        elif any(word in user_input.lower() for word in ["nice chatting to you, bye", "see you till next time", "exit"]):
            st.write("Bye! Have a nice day, come back again soon.")
            break
        # Rule 2: Greeting
        if any(word in user_input.lower() for word in ["hi", "hello", "hey", "good day", "hola"]):
            st.write("Chatbot: Hello, thanks for asking!")
        elif any(word in user_input.lower() for word in ["how are you"]):
            st.write("Chatbot: I am fine, how are you?")
        elif any(word in user_input.lower() for word in ["hi there", "is anyone there?"]):
            st.write("Chatbot: Hi there, how can I help?")
        elif any(word in user_input.lower() for word in ["goodbye", "bye", "see you later", "nice chatting to you"]):
            st.write("Chatbot: Thank you for using the E-commerce Price Comparison Chatbot. Goodbye!")
```

```
        break
    # Rule 2: Providing help
    if any(word in user_input.lower() for word in ["help", "support", "guide"]):
        st.write("Chatbot: I can guide you through different websites and find great offers for the products you want to buy.")
    elif any(word in user_input.lower() for word in ["what you can do", "what help you provide", "what support is offered"]):
        st.write("Chatbot: Offering support to get the best deal out of your purchase.")
    # Rule 3: Websites comparison
    if any(word in user_input.lower() for word in ["websites", "compare with", "how many websites"]):
        st.write("Chatbot: Comparing products from Flipkart, Amazon, and Croma...")
    # Add debug print to check if the user input is received correctly
    print("User input:", user_input)
    # Rule 4: Product comparison
    if any(word in user_input.lower() for word in ["find best deal", "product comparison"]):
        st.write("Sure, please provide me with the product names or keywords you want to compare.")
    elif any(word in user_input.lower() for word in ["compare products", "product comparison", "which one is better", "compare"]):
        product_name = st.text_input("Chatbot: What product would you like to compare prices for?")
        if product_name:
            lowest_price, urls = compare_prices(product_name)  # Modify to unpack only lowest_price and urls
```

```
            print("Lowest price:", lowest_price)
            print("URLs:", urls)
            if lowest_price is not None:
                st.write(f"The lowest price found is ₹{lowest_price}")
                # Display lowest price platform URL
                if urls:
                    st.write("Platform URL:", urls[min(urls, key=urls.get)])
                st.write("")
                st.write("Here are the URLs for your comparison:")
                st.write("Flipkart:", f'https://www.flipkart.com/search?q={product_name.replace("
", "+")}')
                st.write("Amazon:", f'https://www.amazon.in/{product_name.replace(" ", "-
")}/s?k={product_name.replace(" ", "+")}')
                st.write("Croma:", f'https://www.croma.com/search/?text={product_name.replace("
", "-")}')
                st.write("")
            else:
                st.write("No product found.")
            more_queries = st.text_input("Chatbot: Would you like to compare prices for another
product? (yes/no)\n").lower()
            if any(word in more_queries for word in ["no", "bye", "see you later", "goodbye"]):
                # Rule 7
                st.write("Bye! Have a nice day, come back again soon.")
                break
```

```
            elif any(word in more_queries for word in
                ["nice chatting to you, bye", "see you till next time", "exit"]):
                st.write("Thank you for using the E-commerce Price Comparison Chatbot. Goodbye!")
                break
            else:
                st.write("See you!")
        # Add debug print to check if the compare_prices function is being called
        print("End of loop")
        # Rule 5: Thank you
        if any(word in user_input.lower() for word in ["thanks", "thank you", "that's helpful", "awesome, thanks"]):
            st.write("Chatbot: Happy to help!")
# Remember to call the app function to display the app
if __name__ == "__main__":
    app()
```

Comparison:

```
import requests
from bs4 import BeautifulSoup
import re
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
def get_price(url, product_name):
    try:
```

```
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        soup = BeautifulSoup(response.content, 'html.parser')
        product_elements = soup.find_all(lambda tag: product_name.upper() in
tag.text.strip().upper())
        for element in product_elements:
            price_element = element.find_next_sibling(lambda tag: re.search(r'\bprice\b', tag.name,
re.IGNORECASE))
            if price_element:
                return price_element.text.strip().replace(',', '')
        price_elements = soup.find_all(lambda tag: re.search(r'\bprice\b', tag.name,
re.IGNORECASE))
        for element in price_elements:
            price = element.text.strip().replace(',', '')
            if price:
                return price
        return None
    except requests.exceptions.RequestException as e:
        print(f"Error in {url}: {e}")
        return None
def get_flipkart_price(name):
    try:
        name1 = name.replace(" ", "+")
```

```
        url = f'https://www.flipkart.com/search?q={name1}&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=off&as=off'
        res = requests.get(url, headers=headers)
        res.raise_for_status()
        soup = BeautifulSoup(res.text, 'html.parser')
        if soup.select('._4rR01T'):
            flipkart_name = soup.select('._4rR01T')[0].getText().strip().upper()
            if name.upper() in flipkart_name:
                flipkart_price = soup.select('._30jeq3')[0].getText().strip()
                return int(float(flipkart_price.replace("₹", "").replace(",", "")))
        elif soup.select('.s1Q9rs'):
            flipkart_name = soup.select('.s1Q9rs')[0].getText().strip().upper()
            if name.upper() in flipkart_name:
                flipkart_price = soup.select('._30jeq3')[0].getText().strip()
                return int(float(flipkart_price.replace("₹", "").replace(",", "")))
        # Return -1 if the product is not found
        return -1
    except Exception as e:
        print(f"Error in Flipkart: {e}")
        return -1
def get_amazon_price(name):
    try:
```

```
        name1 = name.replace(" ", "-")
        name2 = name.replace(" ", "+")
        url = f'https://www.amazon.in/{name1}/s?k={name2}'
        res = requests.get(url, headers=headers)
        res.raise_for_status()
        soup = BeautifulSoup(res.text, 'html.parser')
        amazon_page = soup.select('.a-color-base.a-text-normal')
        amazon_page_length = len(amazon_page)
        for i in range(amazon_page_length):
            amazon_name = soup.select('.a-color-base.a-text-normal')[i].getText().strip().upper()
            if name.upper() in amazon_name:
                amazon_price = soup.select('.a-price-whole')[i].getText().strip().upper()
                return int(float(amazon_price.replace("₹", "").replace(",", "")))
        # Return -1 if the product is not found
        return -1
    except Exception as e:
        print(f"Error in Amazon: {e}")
        return -1
def get_croma_price(name):
    try:
        name1 = name.replace(" ", "-")
        url = f'https://www.croma.com/search/?text={name1}'
        res = requests.get(url, headers=headers)
```

```
        res.raise_for_status()
        soup = BeautifulSoup(res.text, 'html.parser')
        if soup.select('.product-title'):
            croma_name = soup.select('.product-title')[0].getText().strip().upper()
            if name.upper() in croma_name:
                croma_price = soup.select('.price span')[0].getText().strip()
                return int(float(croma_price.replace("₹", "").replace(",", "")))
        # Return -1 if the product is not found
        return -1
    except Exception as e:
        print(f"Error in Croma: {e}")
        return -1
def compare_prices(product_name):
    try:
        flipkart_price = get_flipkart_price(product_name)
        amazon_price = get_amazon_price(product_name)
        croma_price = get_croma_price(product_name)
        prices = {'Flipkart': flipkart_price, 'Amazon': amazon_price, 'Croma': croma_price}
        if all(price == -1 for price in prices.values()):
            return "No product found on Flipkart, Amazon, or Croma.", {}
        # Filter prices greater than 0
        valid_prices = {key: value for key, value in prices.items() if value is not None and value >
0}
```

```
    if not valid_prices:
        return "No product found on Flipkart, Amazon, or Croma.", {}
    lowest_price = min(valid_prices.values())
    # Create a dictionary with URLs
    urls = {
        'Flipkart': f'https://www.flipkart.com/search?q={product_name.replace(" ", "+")}',
        'Amazon': f'https://www.amazon.in/{product_name.replace(" ", "-")}/s?k={product_name.replace(" ", "+")}',
        'Croma': f'https://www.croma.com/search/?text={product_name.replace(" ", "-")}'
    }
    # Print the minimum price and corresponding URL
    print("_____")
    print(f"\nMinimum Price: ₹{lowest_price}")
    for platform, price in valid_prices.items():
        if price == lowest_price:
            print(f"{platform} URL: {urls[platform]}\n")

    print("----------URLs----------")
    print("Flipkart : \n", urls.get('Flipkart', 'No URL found'))
    print("\nAmazon : \n", urls.get('Amazon', 'No URL found'))
    print("\nCroma : \n", urls.get('Croma', 'No URL found'))
```

```
        print("----------------------------------------------------------------------------------------------------")
        return lowest_price, urls
    except Exception as e:
        print(f"Error in comparing prices: {e}")
        return "An error occurred while comparing prices. Please try again.", {}
def main():
    print("Welcome to the E-commerce Price Comparison Chatbot!")
    while True:
        user_input = input(">> ").strip()
        if any(word in user_input.lower() for word in ["bye", "see you later", "goodbye"]):
            print("Thank you for using the E-commerce Price Comparison Chatbot. Goodbye!")
            break
        elif any(word in user_input.lower() for word in ["nice chatting to you, bye", "see you till next time", "exit"]):
            print("Bye! Have a nice day, come back again soon.")
            break
        if any(word in user_input.lower() for word in ["hi", "hey", "hello", "hola", "good day"]):
            print("Hello, thanks for asking!")
        elif any(word in user_input.lower() for word in ["hi there", "is anyone there?"]):
            print("Hi there, how can I help?")
        elif any(word in user_input.lower() for word in ["how are you"]):
            print("I am fine, how are you?")
```

```
    elif any(word in user_input.lower() for word in ["hi there"]):
        print("Good to see you again!")
    if any(word in user_input.lower() for word in ["how you could help me?", "what you can do?", "what help you provide?", "how you can be helpful?", "what support is offered"]):
        print("I can guide you through different websites and find great offers for the products you want to buy.")
    elif any(word in user_input.lower() for word in ["help", "support", "guide"]):
        print("Offering support to get the best deal out of your purchase.")
    if any(word in user_input.lower() for word in ["websites", "which are the websites do you compare the products with", "which are the websites do you compare with?", "how many website sdo u compare with?", "website names"]):
        print("Comparing products from Flipkart, Amazon, and Croma...")
    if any(word in user_input.lower() for word in ["find best deal"]):
        print("Sure, please provide me with the product names or keywords you want to compare.")
    elif any(word in user_input.lower() for word in ["compare products", "product comparison", "which one is better", "compare"]):
        product_name = input("What product would you like to compare prices for?\n").strip()
        lowest_price, prices = compare_prices(product_name)
        if isinstance(lowest_price, str):
            print(lowest_price)
        else:
            print(f"The lowest price found is ₹{lowest_price}.")
        print("_____")
```

```
        print("Here are the URLs for your comparison:")
        print("Flipkart:", f'https://www.flipkart.com/search?q={product_name.replace(" ", "+")}')
        print("Amazon:", f'https://www.amazon.in/{product_name.replace(" ", "-")}/s?k={product_name.replace(" ", "+")}')
        print ("Croma:", f'https://www.croma.com/search/?text={product_name.replace(" ", "-")}')
        print("_____")
        more_queries = input("Would you like to compare prices for another product? (yes/no)\n").lower()
        if any(word in more_queries for word in ["no", "bye", "see you later", "goodbye"]):
            print("Bye! Have a nice day, come back again soon.")
            break
        elif any(word in more_queries for word in ["nice chatting to you, bye", "see you till next time", "exit"]):
            print("Thank you for using the E-commerce Price Comparison Chatbot. Goodbye!")
            break
        else:
            print("See you!")
    if any(word in user_input.lower() for word in ["thanks", "thank you", "that's helpful", "awesome, thanks", "thanks for helping me"]):
        print("Happy to help!")
if __name__ == "__main__":
    main()
```

CHAPTER 6
RESULT

RESULT

The project was undertaken with the objective of creating a comprehensive comparison website that could gather data from multiple sources and effectively compare prices across various platforms. The primary focus was to provide users with a streamlined interface where they could easily view and compare prices, ultimately enabling them to make informed purchasing decisions based on the lowest available price.

Fig 5.1 Home page

Fig 5.2 About page

The Price Probe project offers a comprehensive solution for online shoppers, seamlessly integrating various features to enhance the shopping experience. The login page(Fig 5.3) serves as the secure entry point for registered users, ensuring privacy and control over user-specific data and functionalities. Once logged in, users are greeted by the central hub of the platform—the home page(Fig.5.1), which provides a user-friendly interface for easy navigation and access to essential features such as product searches, price comparisons, and personalized recommendations

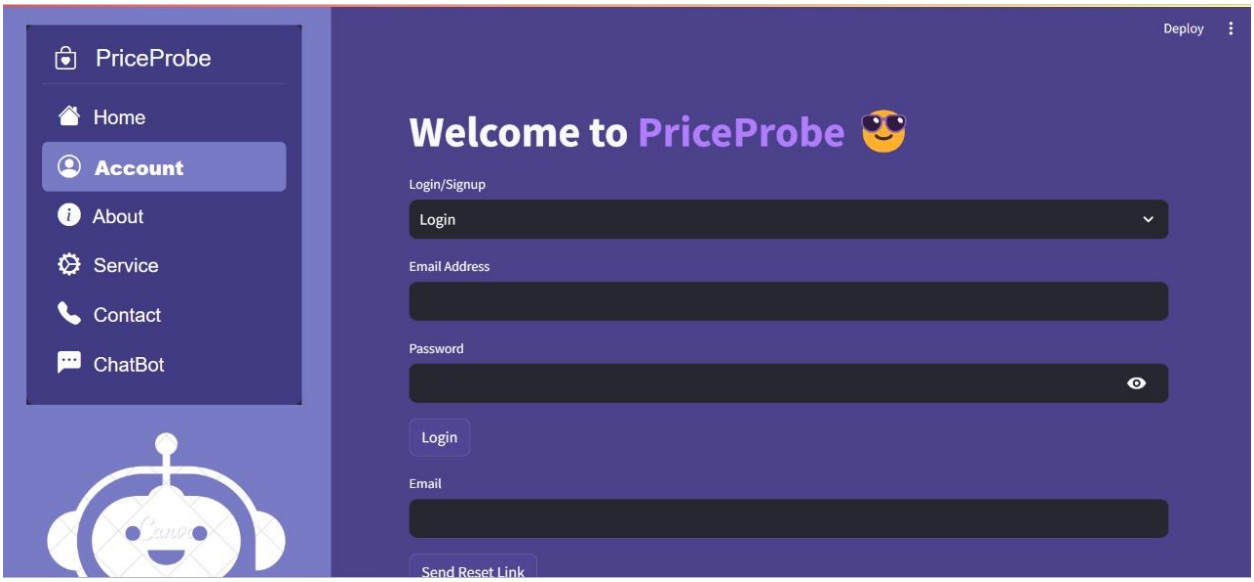


Fig 5.3 Login Page

Additionally, the about page (Fig 5.2) offers users detailed insights into the Price Probe project, including its objectives, mission statement, and the dedicated team behind its development. This transparency fosters trust and confidence among users, enhancing their overall experience on the platform. The chatbot (Fig 5.4) feature further elevates user engagement by serving as a virtual assistant, offering personalized recommendations, answering queries, and providing assistance throughout the shopping journey.

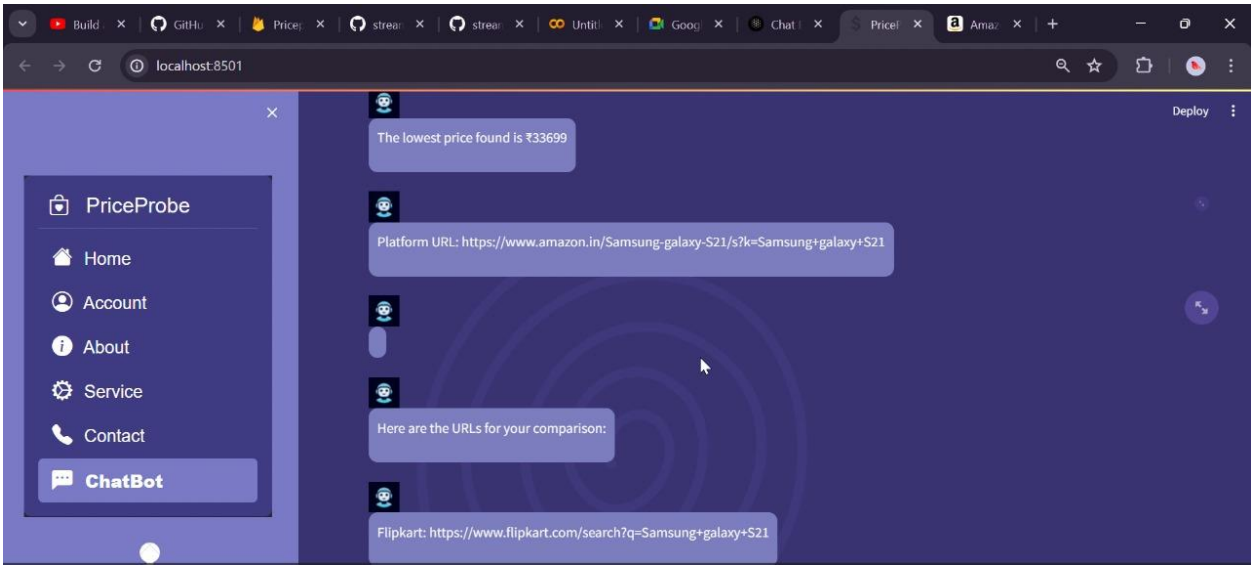


Fig 5.4 Chatbot

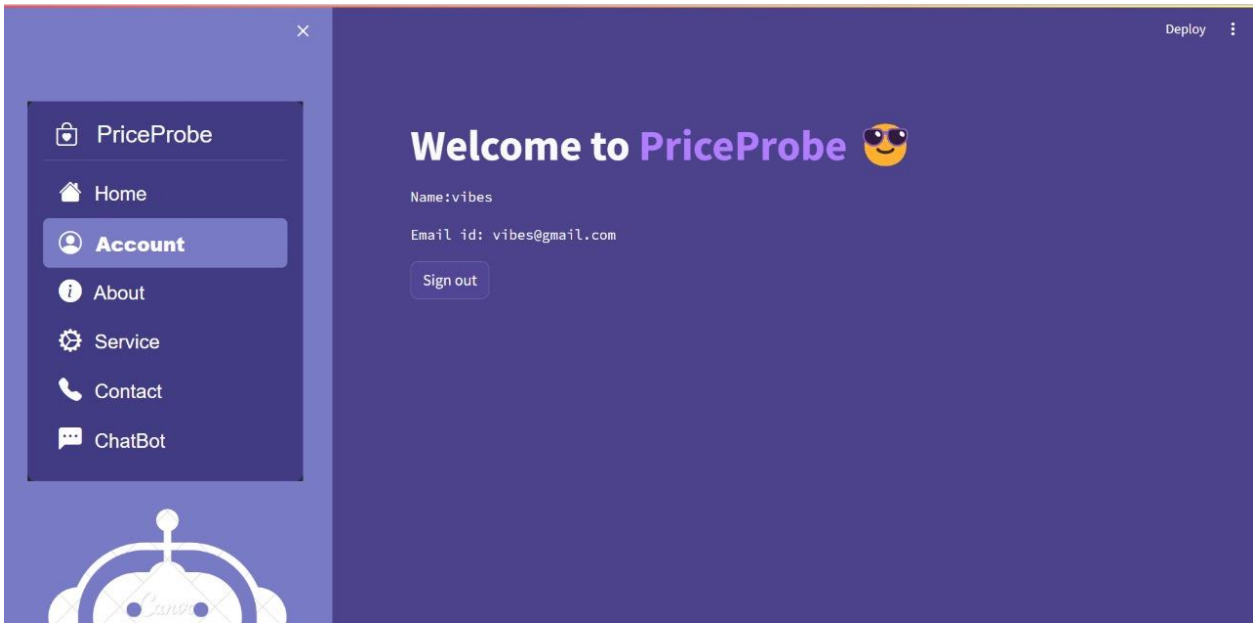
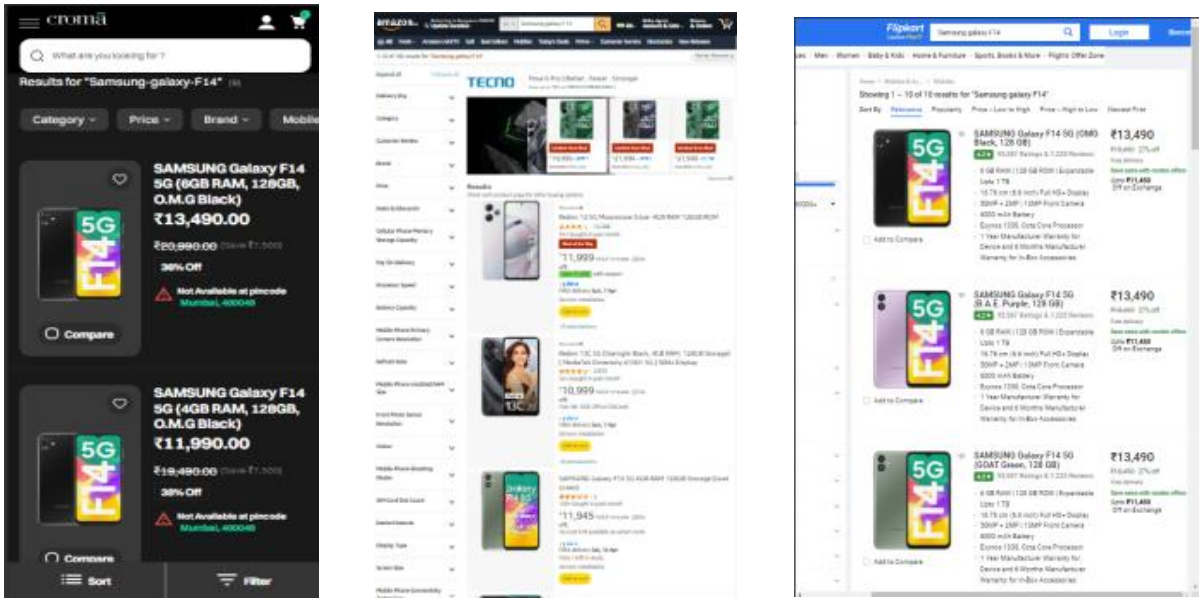


Fig 5.5 Sign out

Furthermore, the comparison tool empowers users to make informed decisions by enabling them to compare prices, specifications, and other relevant details of products across multiple e-commerce platforms such as Flipkart, Amazon and Croma (Fig 5.6). Presented in a clear and structured manner, this feature facilitates easy comparison and aids users in selecting the best option for their needs. Finally, the lowest price feature ensures that users can swiftly identify the

most cost-effective deals available, saving time and maximizing savings during their online shopping endeavors. Together, these features culminate in a platform that prioritizes user convenience, transparency, and savings, ultimately enhancing the overall online shopping experience.



(a) (b) (c)

Fig 5.6 Multichannel Shopping with Price Comparison

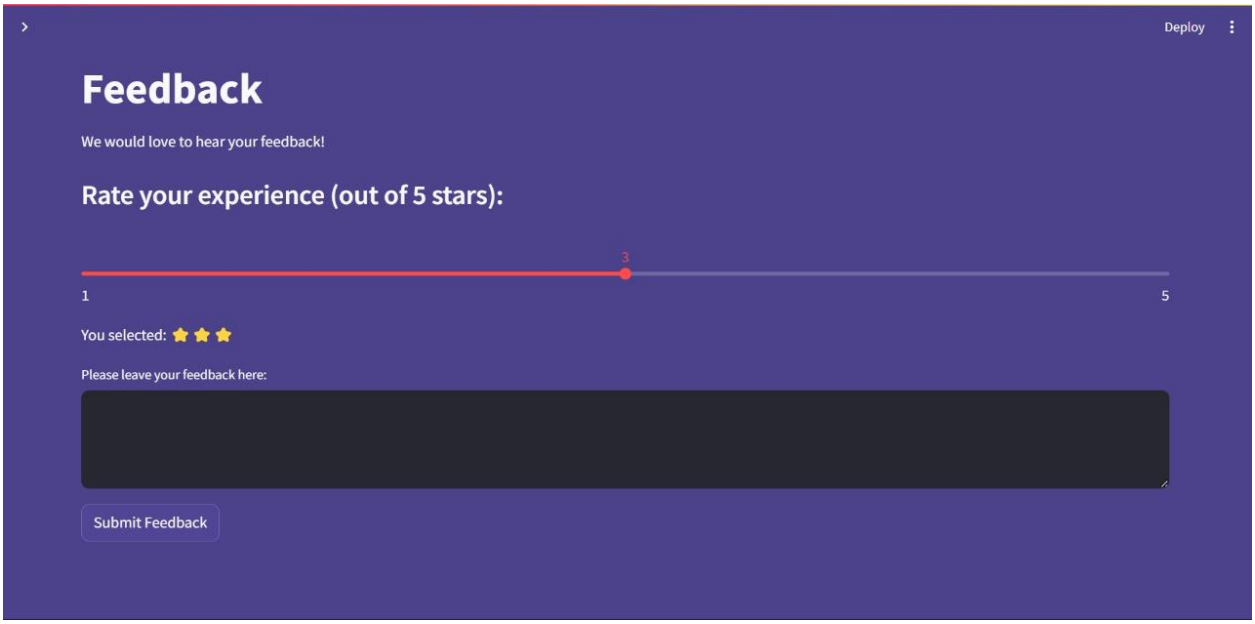


Fig 5.6 Feedback

Hardware and Software Requirements

Table I. System Configuration Details

Hardware Requirements	• A computer with a multi-core CPU • High-end graphics card (GPU) • Sufficient RAM would be required to train deep learning models on large datasets
Software Requirements	• Windows 32/64- bit operating System
Platform	• Windows 32/64-bit operating System
Programming Language/Tools	• Python • TensorFlow • Keras • Jupyter notebook • Beautifulsoup • NLTK • Pickel • Streamlit

CONCLUSION

The innovative chatbot simplifies online shopping through comprehensive comparisons and exclusive discounts from various platforms, revolutionizing the e-commerce landscape. By leveraging advanced algorithms and real-time data analysis, it offers personalized recommendations, learning from user interactions to enhance relevance and responsiveness. Integrated with Streamlit and Natural Language Processing, the chatbot provides a user-friendly interface and interprets user inputs accurately. With functionalities like price retrieval, personalized recommendations, and comparison, users make informed decisions, maximizing convenience and savings. Future work includes implementing image recognition for users to search products by sharing images, enhancing the chatbot's capabilities and appeal.

REFERENCES

[1] Shalini, A., and Ambikapathy, R. "E-Commerce Analysis and Product Price Comparison Using Web Mining." International Journal of Research Publication and Reviews, vol. 3, no. 6, June 2022, pp. 3620-3623. ISSN: 2582-7421.

[2] O. S. Al-Mushayt, W. Gharibi and N. Armi, "An E-Commerce Control Unit for Addressing Online Transactions in Developing Countries: Saudi Arabia—Case Study," in IEEE Access, vol. 10, pp. 64283-64291, 2022, doi: 10.1109/ACCESS.2022.3180329.

[3] Shaikh, A., Sonmali, A., & Wakade, S. (2023). Product Comparison Website using Web scraping and Machine learning. International Research Journal of Engineering and Technology (IRJET), 10(11), 573. https://doi.org/10.2395/0056-0072.573

[4] L. Beranek and R. Remes, "E-commerce network with price comparator sites," 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic, 2019, pp. 401-404, doi: 10.1109/ACITT.2019.8779865.

[5] Shaikh, Arman & Khan, Raihan & Panokher, Komal & Ranjan, Mritunjay & Sonaje, Vaibhav. (2023). E-commerce Price Comparison Website Using Web Scraping. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences. Volume 11. 1-13. 10.37082/IJIRMPS.v11.i3.230223.

[6] Bhabad, H.P., Vyavahare, Atharva, Dengale, Abhishek, Hiwale, Yogesh, and Gosavi, Mehul. "BEST PRICE - PRODUCT COMPARISON ANDROID APP FOR ONLINE AND OFFLINE MARKET." International Research Journal of Modernization in Engineering Technology and Science, vol. 05, no. 05, May 2023, p. 5548. e-ISSN: 2582-5208.

[7] Vasudevan, Srividhya & Megala, P.. (2019). Scraping and Visualization of Product Data from E-commerce Websites. International Journal of Computer Sciences and Engineering. 7. 1403-1407. 10.26438/ijcse/v7i5.14031407.

[8] Maurya, H., Patil, K., Sawant, S., Thange, M., & Mahadik, A. (2023). Price Comparison Website. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 3(2).

[9] Sowmiya, M., Srinandhan, CS., Mugesh Raja, M., & Sudheekshan Kumar, S. (2023). Price Comparison for Products in Various ECommerce Website. International Journal for Research Trends and Innovation, 8(5), 570. ISSN: 2456-3315.

[10] G. A. Rao, K. Syamala, P. V. V. Kishore and A. S. C. S. Sastry, "Deep convolutional neural networks for sign language recognition," 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES), Vijayawada, India, 2018, pp. 194-197, doi: 10.1109/SPACES.2018.8316344.

[11] A. Chandrasekhar, L. Singh, S. Tripathi and N. Malik, "Sign Language Recognition System Using Deep Learning," 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2023, pp. 67-71, doi: 10.1109/Confluence56041.2023.10048804.

[12] S. Almana and A. Al-Omary, "Real-time Arabic Sign Language Recognition using CNN and OpenCV," 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakheer, Bahrain, 2022, pp. 32-36, doi: 10.1109/3ICT56508.2022.9990643.