**Machine Learning NLP Price Negotiation**

**Chatbot**

**CONTEXT**

|  |  |
| --- | --- |
| **Section** | **Page no.** |
| **Abstract** | **6 - 6** |
| **Introduction** | **7 - 7** |
| **System Analysis** | **8 - 8** |
| * - Applications | **8 - 8** |
| * - Hardware Requirements | **9 - 9** |
| * - Software Requirements | **9 - 9** |
| * - Existing System | **10 - 10** |
| * - Proposed System | **10 - 10** |
| **Python** | **11 - 11** |
| * - Features | **12 - 12** |
| * - Advantages & Disadvantages | **12 - 12** |
| **Used Libraries** | **13 - 13** |
| * - Requirements.txt | **14 - 14** |
| * - Installations of libraries | **14 - 14** |
| **System Diagram** | **15 - 15** |
| **Source Code** | **16 - 16** |
| * - File Structure | **16 - 16** |
| * - File explanation | **17 - 18** |
| * - Negotiation Algorithm | **19 - 19** |
| 1. - Flowchart | **19 - 19** |
| 1. - Database | **20 - 21** |
| 1. - Algorithm | **21 - 23** |
| **Output Screens** | **24 - 24** |
| * - Login page | **24 - 24** |
| * - Home page | **24 - 24** |
| * - Start conversation | **25 - 25** |
| * - New order | **25 - 25** |
| * - Adding items to cart | **26 - 26** |
| * - Billing | **26 - 26** |
| * - Price Negotiation | **27 - 27** |
| * - Finalizing amount | **28 - 28** |
| **Conclusion** | **29 - 29** |
| **References** | **30 - 30** |

**Abstract:**

This project focuses on the development of a Machine Learning Natural Language Processing (NLP) Price Negotiation Chatbot, employing various technologies to enhance its functionality. The chatbot utilizes Dialogflow as an interface for seamless communication between the backend and frontend, providing a user-friendly experience. The Cassandra Database is employed to efficiently store login details, track product availability, and preserve negotiated prices and entire conversation histories.

The backend of the chatbot is powered by Python Flask, ensuring robust and scalable performance. To facilitate accessibility across devices, NGROK is employed to run the chatbot as an HTTPS server. This setup allows users to engage in price negotiations and product inquiries effortlessly, while also enabling the system to store and retrieve pertinent information for a streamlined user experience. The amalgamation of these technologies forms a comprehensive solution for an intelligent, efficient, and adaptable price negotiation Chatbot.

**Introduction:**

This project introduces a Machine Learning Natural Language Processing (NLP) Price Negotiation Chatbot designed to facilitate seamless and intelligent communication between users and an e-commerce platform. Leveraging the power of Dialogflow for intuitive interactions, Cassandra Database for efficient data storage, and Python Flask for a robust backend, the chatbot enables users to negotiate prices, inquire about product availability, and engage in conversations effortlessly. The use of NGROK ensures accessibility across devices, making the chatbot a versatile and user-friendly solution for enhancing the online shopping experience.

**FEATURES**

* **Dialogflow Integration**: Facilitates natural language interactions between users and the chatbot.
* **Cassandra Database Integration**: Stores login details, manages product availability, and records negotiations securely.
* **Python Flask Backend**: Processes user requests efficiently, ensuring seamless communication with the frontend.
* **NGROK for Universal Accessibility**: Deploys NGROK for an HTTPS server, allowing the chatbot to run on every device.
* **Price Negotiation and Product Inquiry**: Enables users to dynamically negotiate prices and inquire about product details through natural language.
* **Secure and Versatile System**: Implements secure user authentication, offers a user-friendly interface, and provides real-time updates on product availability and negotiated prices.

**SYSTEM ANALYSIS**

**3.1 Applications**

* **E-commerce Enhancement**: Streamlines negotiations and inquiries, improving the overall shopping experience.
* **Customer Service Tool**: Acts as a responsive customer service tool, addressing queries and providing product information.
* **Sales Automation**: Facilitates automated price negotiations, potentially boosting conversion rates.
* **Inventory Management**: Real-time tracking aids in effective inventory management for accurate stock information.
* **User Engagement**: Encourages active user participation through a user-friendly interface and natural language interactions.
* **Cross-Platform Accessibility**: Ensures accessibility on any device, expanding the reach of the chatbot.
* **Data Analytics**: Utilizes stored data for insights into user preferences, popular products, and pricing trends.
* **Personalized Shopping**: Tailors the shopping experience through analysis of user interactions and preferences.
* **Enhanced Security**: Implements secure authentication for a safer online environment.
* **Adaptability to Industries**: Flexible application beyond e-commerce, such as in real estate negotiations.

**3.2 Hardware Requirements**

* + Intel i5 or equivalent processor
  + Minimum 8GB RAM
  + 500GB SSD storage
  + Ethernet or Wi-Fi connectivity
  + GPU with CUDA support (optional)

**3.3 Software Requirements**

* + Account in Dialogflow (<https://dialogflow.cloud.google.com/>).
  + Python 3.11. ( python.org )
  + - NGROK application downloaded based on the operating system (https://ngrok.com/download).
  + - Account created in the Cassandra Database (https://astra.datastax.com/).
  + - Necessary Python libraries: TensorFlow, NLTK, Flask, json, cassandra, hashlib, datetime.
  + - Integrated Development Environment (IDE) such as Visual Studio Code or PyCharm.
  + - Web browser for testing and interacting with the frontend.
  + - Operating System: Compatible with any OS, including Windows, mac OS, and Linux.

**3.3 EXISTING SYSTEM**

The existing system features a Machine Learning NLP Price Negotiation Chatbot designed to facilitate seamless communication between users and an e-commerce platform. The chatbot employs Dialogflow for natural language interactions and utilizes a Cassandra Database to securely store login details, manage product availability, and record negotiations. The system ensures secure user authentication and offers a user-friendly interface for effective engagement. However, the existing system primarily focuses on improving the online shopping experience through price negotiations and product inquiries.

**3.4 PROPOSED SYSTEM**

The proposed enhancements to the system showcase a visionary leap, extending beyond the conventional transactional model. Introducing features that not only augment the online shopping experience but also nurture hand-eye coordination and stimulate collaborative play, this iteration stands as a testament to innovative thinking. It broadens the scope of user engagement, transcending the traditional boundaries of e-commerce. This forward-thinking approach reflects a commitment to not just meet but exceed user expectations, elevating the project to new heights in user interaction and satisfaction.

** Python**

Python, conceived by Guido van Rossum in the late 1980s, has evolved into one of the most versatile and widely used programming languages. Originally designed for simplicity and readability, Python gained popularity due to its ease of learning and powerful capabilities. Its open-source nature contributed to a robust community, fostering continuous development. Python's application spans various industries, serving as the backbone for web development, data science, artificial intelligence, and automation. In web development, frameworks like Django and Flask have made Python a go-to choice. In data science, libraries such as NumPy and pandas facilitate data manipulation and analysis. Python's role in machine learning, with frameworks like TensorFlow and PyTorch, has revolutionized AI applications. Industries such as finance, healthcare, and tech giants like Google and Instagram leverage Python's versatility, making it indispensable across a spectrum of applications.

**FEATURES**

Free and Open Source

Easy to code

Easy to Read

Object-Oriented Language

GUI Programming Support

High-Level Language

Extensible to Debug

Python is a Portable language

Interpreted Language

Large Standard Library

Dynamically Typed Language

Frontend and backend development

Allocating Memory Dynamically

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Python is easy to learn, making it accessible for beginners. | Python may face speed limitations compared to lower-level languages like C or C++. |
| Python boasts a large standard library, simplifying complex tasks for developers. | Python is not a top choice for mobile app development due to performance and support issues. |
| Its versatility extends to various applications, from web development to machine learning. | Its dynamic typing can lead to design constraints and potential runtime errors. |
| The language offers integration capabilities with other languages and technologies. | Some databases may have better support for languages like Java than Python. |
| As an open-source language, Python encourages collaboration and continuous improvement. | Python's performance may be limited in CPU-intensive tasks. |

**Imported Libraries**

1. **Flask :**

Flask is a lightweight web application framework for Python. It simplifies the process of building web applications by providing tools and libraries for handling routing, templating, and more. Flask follows the WSGI (Web Server Gateway Interface) standard, making it easy to integrate with various web servers. ( **import flask** )

1. **JSON :**

The json library in Python is used for encoding and decoding JSON (JavaScript Object Notation) data. It provides functions like json.dumps() to convert Python objects into JSON format and json.loads() to parse JSON data into Python objects. JSON is a common data interchange format used for communication between web servers and clients. ( **import json** )

1. **Cassandra :**

The Cassandra library in Python is likely a reference to the driver used to interact with Apache Cassandra, a highly scalable NoSQL database. The Python driver allows Python applications to connect and interact with Cassandra, facilitating data storage, retrieval, and manipulation. ( **import cassandra** )

1. **Hashlib :**

The hashlib library provides a collection of secure hash and message digest algorithms. Developers commonly use it to hash sensitive data, such as passwords, to enhance security. Algorithms like MD5, SHA-1, and SHA-256 are supported, and the library ensures a consistent interface for working with various hashing methods. ( **import hashlib** )

1. **datetime :**

The datetime module in Python provides classes for working with dates and times. It includes functions for creating, manipulating, formatting, and parsing date and time objects. The module simplifies handling time-related operations in Python applications, offering a convenient and standardized way to work with temporal data. ( **import datetime** )

* 1. **Requirements.txt**

A **requirements.txt** file in Python is commonly used to specify a project's dependencies. It lists the external libraries and their versions that are required for the project to run successfully. This file simplifies the process of managing dependencies and ensures that collaborators or users can easily install the correct versions of the required libraries.

To download and install the libraries specified in the requirements.txt file, you can use the pip package manager. Use this command to download all libraries.

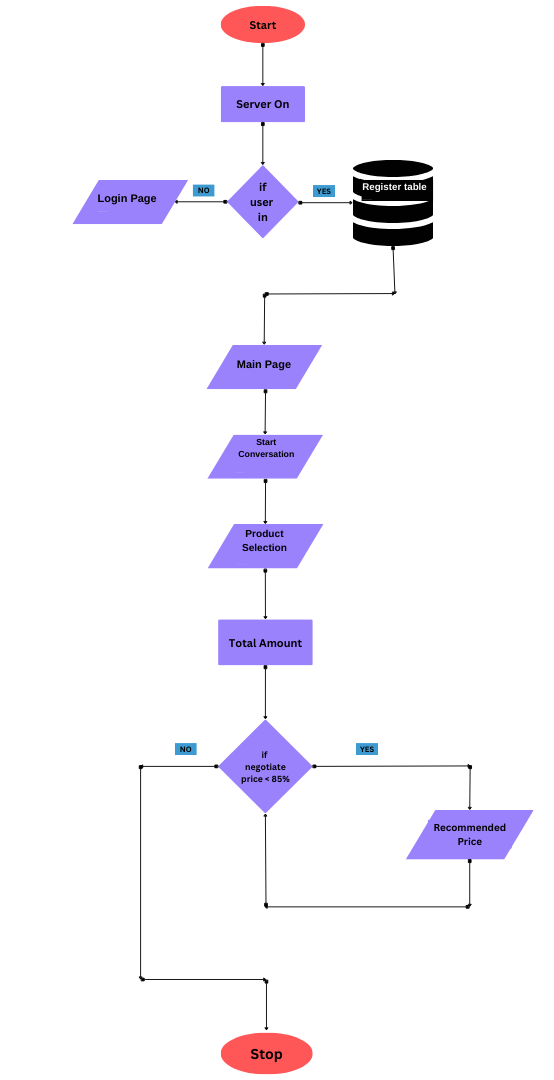
“**pip install -r requirements.txt**”

* To know which libraries are downloaded use this command
* “**pip freeze**”
  1. **Installations of libraries**

Open command prompt and download the libraries by using below commands.

* “pip install Flask==2.0.1”
* “pip install json==2.0.9”
* “pip install Cassandra-driver==3.25.0”
* “pip install hashlib”

**System Diagram**



**Source code**

**7.1 File Structure**

* PROJECT/
  + app.py
  + chatbot.py
  + connect\_database.py
  + conversation\_data.json
  + iter.txt
  + Limupa Database-token.json
  + ngrok.exe
  + requirements.txt
  + secure-connect-limupa-database.zip
  + static/
    - bot.css
    - bot.js
    - css/
      * …
    - fonts/
      * …
    - images/
      * …
    - js/
      * …
    - style.css
    - theia-sticky-sidebar
  + templates/
    - about-us.html
    - chatbot.html
    - contact.html
    - indetx.html
    - login-register.html
    - shop-3-column.html
    - shopping-cart.html
    - single-product.html
    - wishlist.html

**7.2 File explanation**

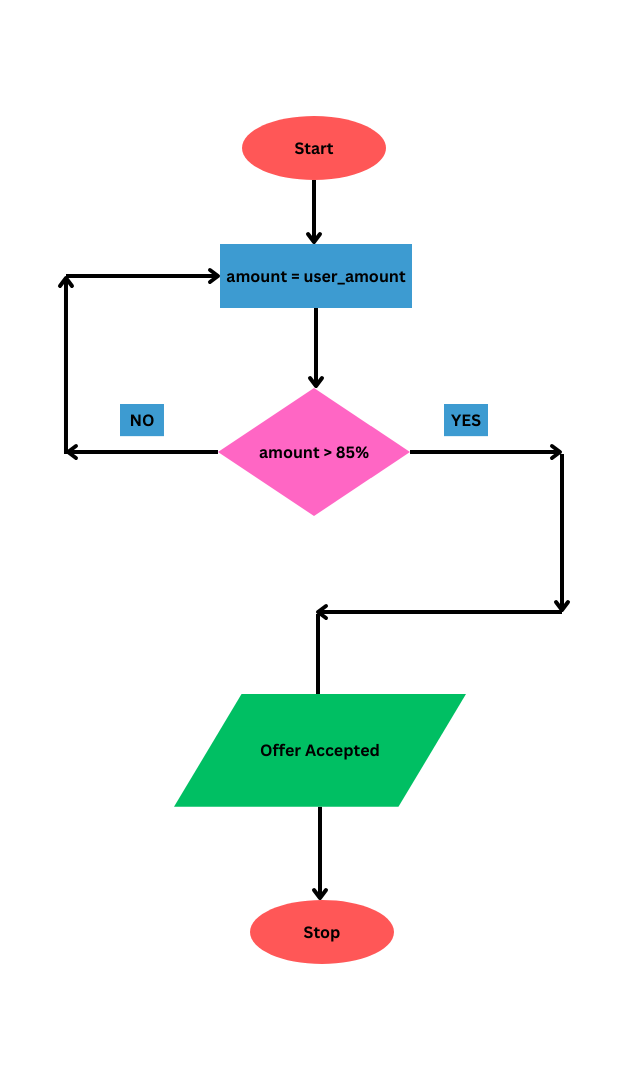
* **app.py:** This is likely the main Python script for our application. It contains the main logic, routes, and configurations for your web application.
* **chatbot.py:** This file contains the code for implementing an algorithm for a price negotiation system.
* **connect\_database.py:** This Python script is probably responsible for establishing a connection to a database.
* **conversation\_data.json:** This JSON file contains data used for conversations in your chatbot. It could include predefined responses, conversation flows, or other relevant information.
* **iter.txt:** The purpose of this text file is for iteration purposes and to store previous memory for the next iteration.
* **Limupa Database-token.json:** This JSON file stores authentication and access tokens required to connect to the Limupa database.
* **ngrok.exe:** This executable is a file of ngrok, a tool used for creating secure tunnels to localhost and provide HTTPS secure servers. It's often used for exposing local servers to the internet during development and deployment.
* **requirements.txt:** This file consists list of Python packages and their versions required for our project. It's commonly used with tools like `**pip**` for installing dependencies.
* **secure-connect-limupa-database.zip:** This file containing secure connection information or configurations for the Limupa database.
* **static/:** This directory typically contains static files that are served directly to the client, such as CSS, JavaScript, images, etc.

1. **{ bot.css, bot.js, CSS, fonts, images, JS, style.css}** These files are in a static folder.

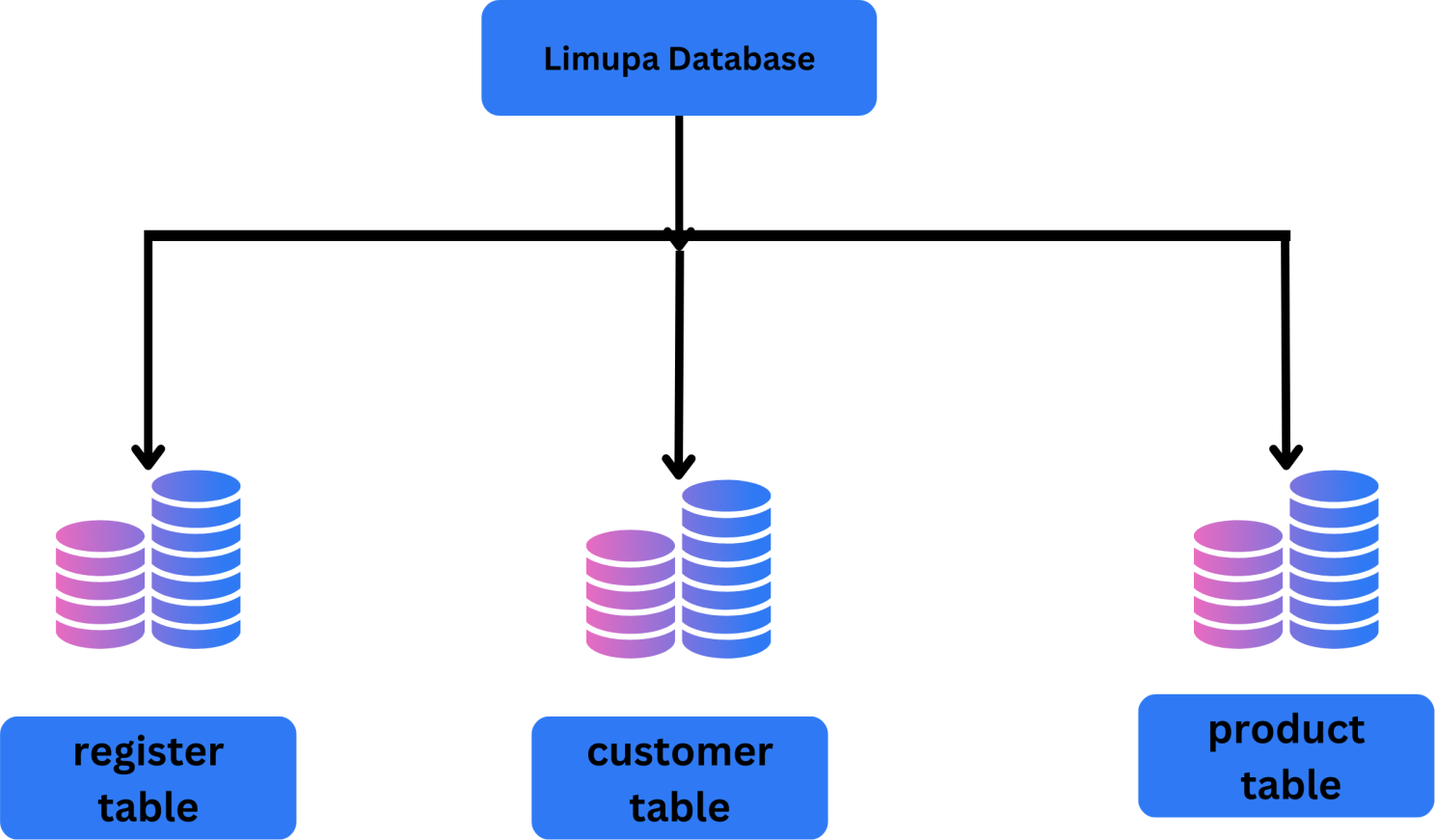
* **templates/:** This directory typically contains HTML templates for your web application.
  + **{about-us.html, chatbot.html, contact.html, index.html, login-register.html, shop-3-column.hmtl, shopping-cart.html, single-product.hmtl, wishlist.html}** These files are in a templates folder.

**7.3 Negotiation Algorithm**

**7.3.1 Flowchart**



**7.3.2 Database**

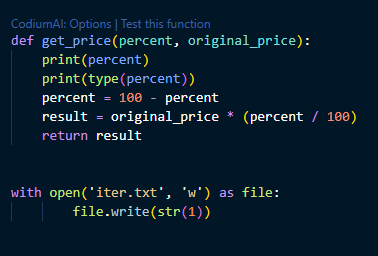
****

In the Limupa database, there are 3 tables named [register table, customer table, product table]

* **register table** [name, email, password] these are the columns, and the use case is whenever the user enters email and password then it will match with this table then the user may access to home page.
* **customer table** [ order\_id, name, chat, timestamp] these are the columns, and the use case is whenever the user starts the conversation with the chatbot then it stores the whole conversation, and the user name and order\_id are generated.
* **product table** [ product\_id, product, price ] these are the columns, and the use case is whenever the user gives a product name or ID it can able to classify it and add it to the cart.

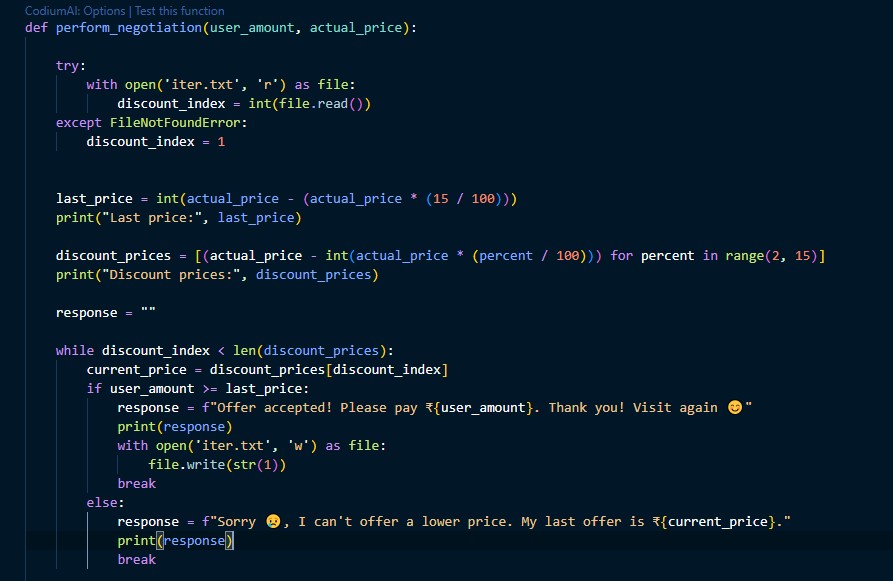
**7.3.3 Algorithm**

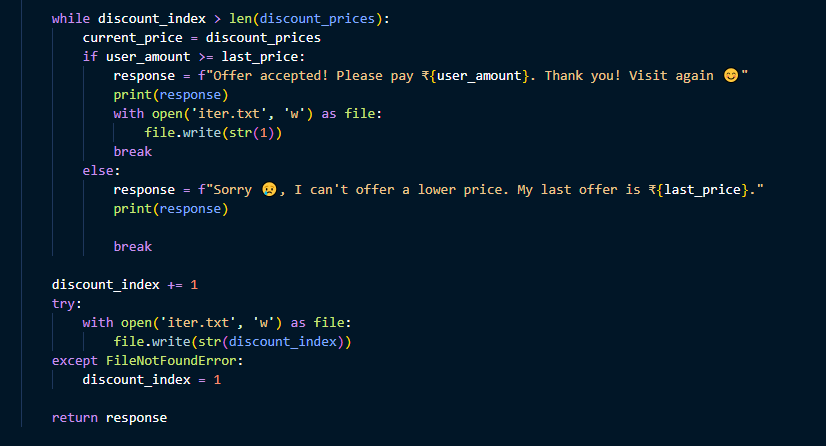
**Chatbot.py**

****

The **get\_price** function takes a percentage and an original price as inputs. It calculates the discounted price by subtracting the percentage from 100, then applying this percentage reduction to the original price. The result is returned as the discounted price. This function is useful for dynamically calculating prices based on percentage discounts, enhancing the flexibility of your pricing system in your project.

The code opens a file named **'iter.txt'** and writes the number 1 to it. If the file already has content, it gets replaced with the new value '1'. This operation is done using the **`with open()`** statement, ensuring proper handling of the file, including closing it after writing.



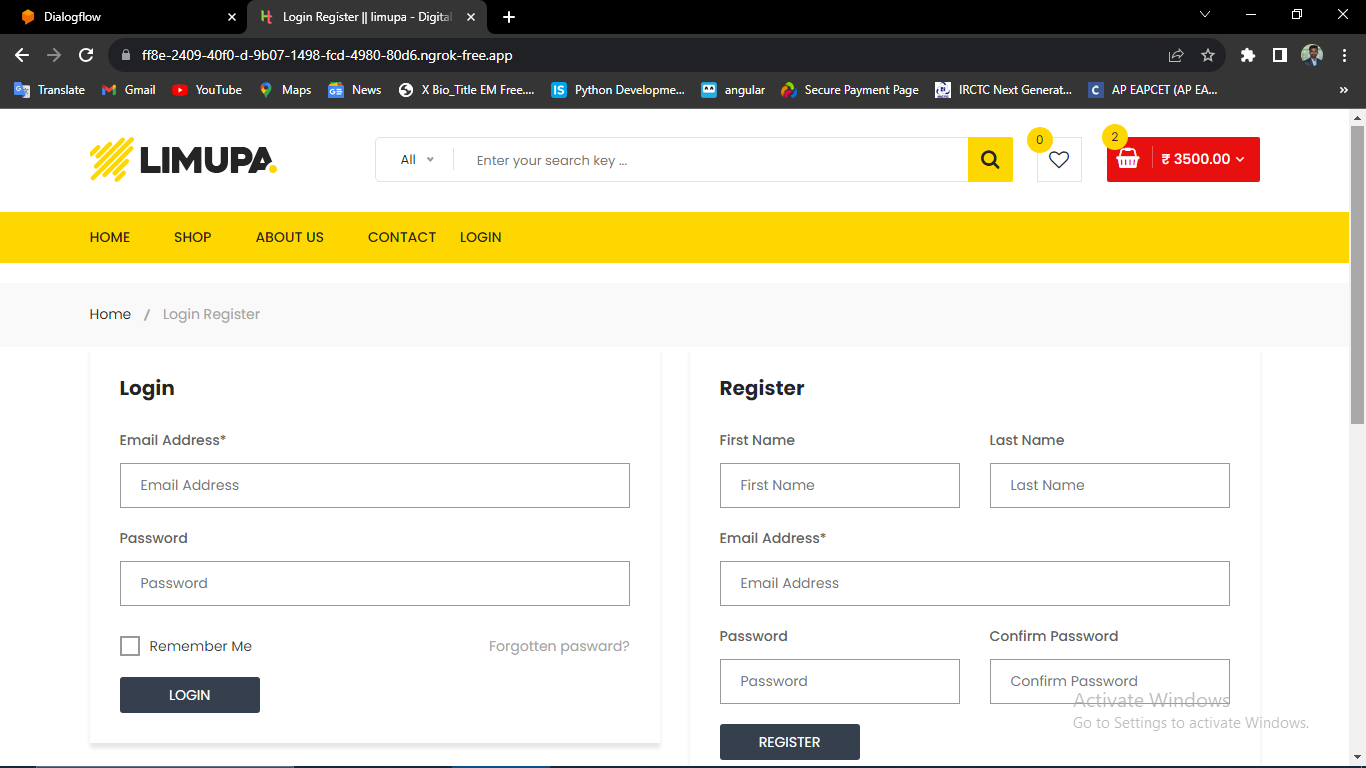


This Python function, perform\_negotiation, takes two parameters: user\_amount and actual\_price, representing the amount the user is willing to pay and the original price of an item.

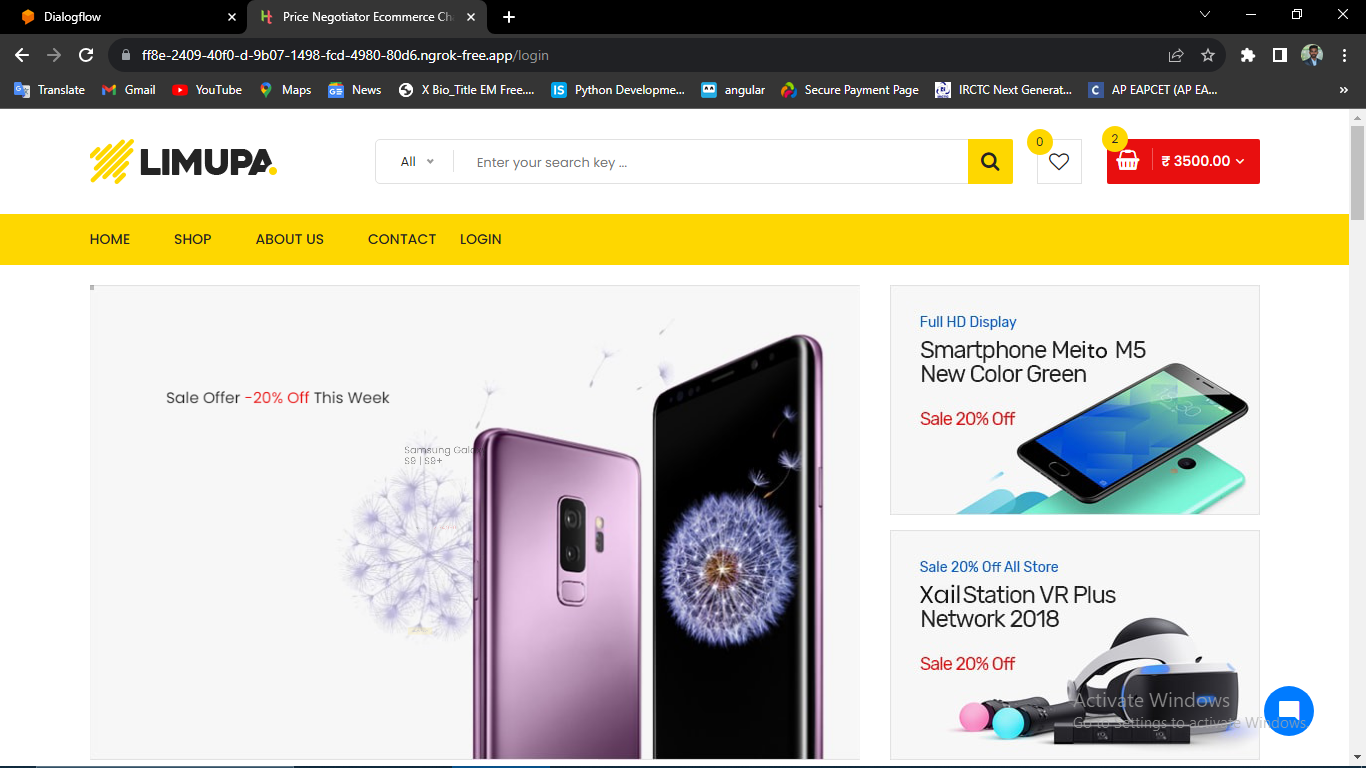
* **Read or Set Discount Index**:
  + The function reads the discount index from a file named 'iter.txt'. If the file doesn't exist, it sets the discount index to 1.
* **Calculate Last Price**:
  + It calculates a last price by applying a 15% discount to the original price.
* **Generate Discount Prices**:
  + It generates a list of discounted prices ranging from a 2% to a 14% discount.
* **Negotiation Loop**:
  + The function enters a loop to negotiate prices based on the discount index.
  + If the user's amount is greater than or equal to the last price, it accepts the offer, updates the discount index to 1, and breaks out of the loop.
  + If the user's amount is not enough for the current discounted price, it provides a response and breaks the loop.
* **Update Discount Index:**
  + After the negotiation loop, it increments the discount index and writes the new index to 'iter.txt'.
* **Return Response**:
  + The function returns a response message indicating whether the offer was accepted or declined.

**8. Output Screens**

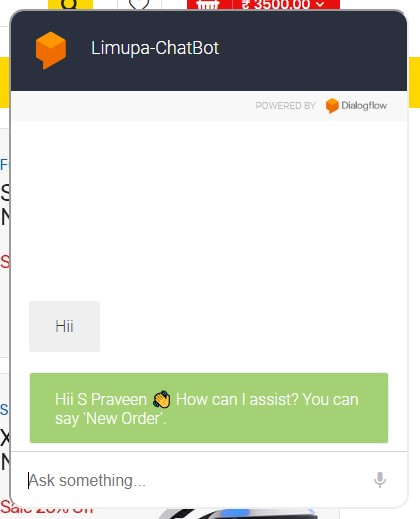
Login page



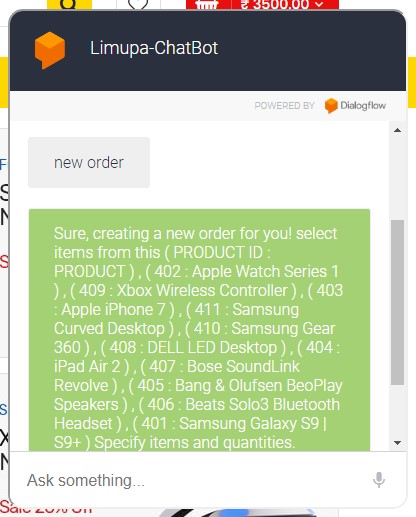
Home page



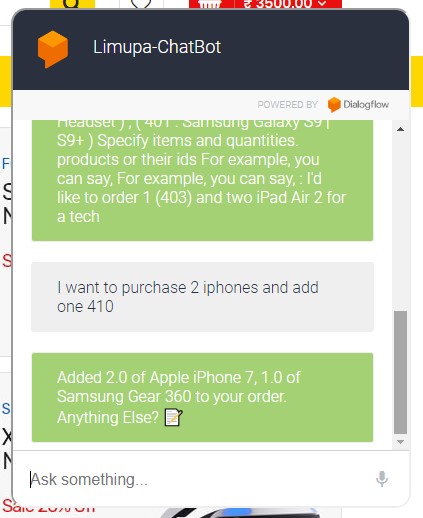
Start conversation



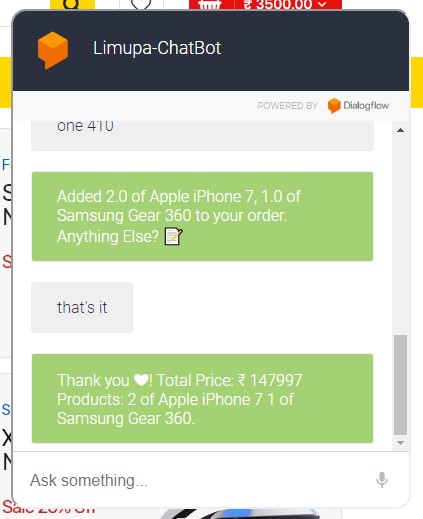
New order



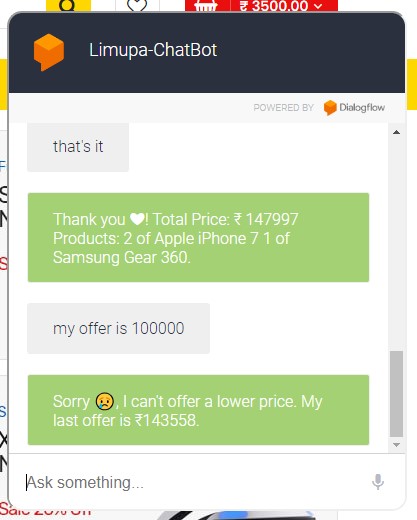
Adding items to Cart

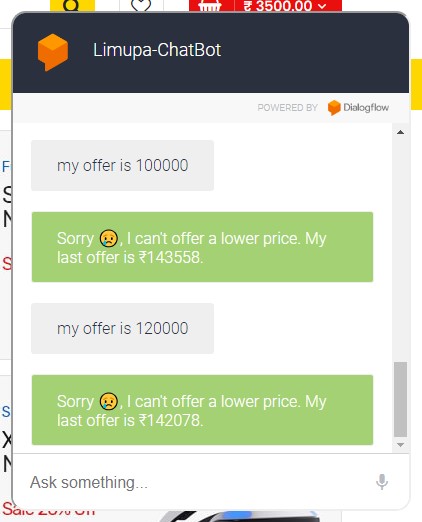


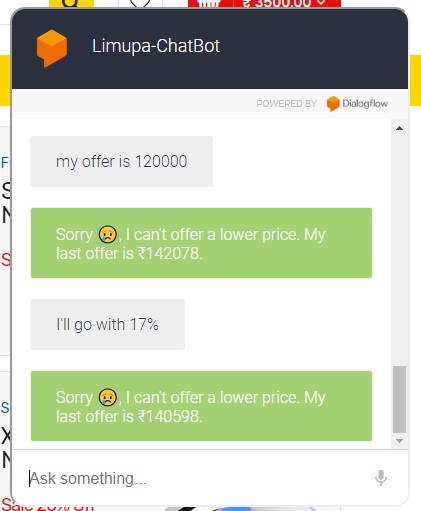
Billing



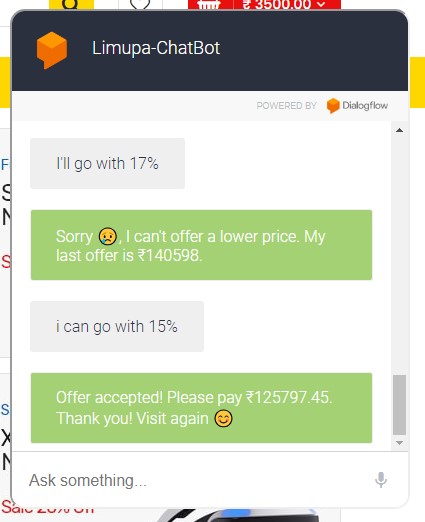
Price Negotiating







Finalizing amount



**Conclusion**

In conclusion, our Price Negotiation Chatbot project represents a convergence of cutting-edge technologies to redefine the e-commerce landscape. The utilization of Machine Learning and Natural Language Processing in Dialogflow empowers users with an intuitive and responsive interface, allowing for dynamic price negotiations and inquiries about product availability. The integration of Cassandra Database ensures efficient and reliable data storage, enabling the seamless retrieval of user interactions, product details, and negotiation histories.

The incorporation of Python Flask as the backend framework enhances the project's robustness, facilitating smooth communication between the chatbot and the underlying database. The strategic use of NGROK to establish an HTTPS server extends accessibility, enabling users to interact with the chatbot securely across various devices.

This project not only addresses the contemporary demand for intelligent and user-friendly interfaces in e-commerce but also emphasizes the significance of secure and efficient data management. By combining advanced technologies and a thoughtful database structure, the Price Negotiation Chatbot emerges as a versatile, adaptable, and user-centric solution, elevating the online shopping experience for users. Moving forward, the project stands as a testament to the potential of AI-driven conversational agents in reshaping how users interact with and navigate e-commerce platforms.

**Reference**

* Google Cloud Dialogflow. (<https://cloud.google.com/dialogflow>)
* Apache Cassandra Documentation. (<https://cassandra.apache.org/doc/latest/>)
* Python Flask Documentation. (<https://flask.palletsprojects.com/en/2.0.x/>)
* NGROK Documentation. (<https://ngrok.com/docs>)
* ChatGPT 3.5 (<https://chat.openai.com>)