



Food Seeker AI– Chat Using Python and Flask:

Shivalingu K G¹, Vishwas S², Mr. Subhash Kamble³

¹²Students, Department of Information and Science Engineering.

³Assistant Professor, Department of Information and Science Engr.

Global Academy of Technology, Bengaluru, Karnataka.

Abstract: This project introduces Voxbot Assistant, an artificial intelligence (AI) chatbot built with Flask that aims to improve user engagement by leveraging speech recognition and natural language processing. By reacting to different prompts with pre-programmed responses, answering frequently asked questions, and providing useful links, the chatbot seeks to deliver an entertaining and user-friendly experience. To enable hands-free operation, Voxbot Assistant also has a speech recognition feature that lets users communicate with the chatbot by speaking orders to it. The aesthetically pleasing and user-friendly interface, constructed using HTML, CSS, and JavaScript, guarantees a positive user experience. In order to demonstrate Voxbot Assistant's possible uses in customer service, personal help, and educational tools, this paper examines the architecture, development process, and features of the tool. The project shows how web technology and AI may be combined to create an interactive and flexible.

Keywords: Flask, Chatbot Interface, Autocomplete ,Responses, User Experience , Web applications.

I. Introduction

This project introduces Voxbot Assistant, an artificial intelligence (AI) chatbot built with Flask that aims to improve user engagement by leveraging speech recognition and natural language processing. By reacting to different prompts with pre-programmed responses, answering frequently asked questions, and providing useful links, the chatbot seeks to deliver an entertaining and user-friendly experience. To enable hands-free operation, Voxbot Assistant also has a speech recognition feature that lets users communicate with the chatbot by speaking orders to it. The aesthetically pleasing and user-friendly interface, constructed using HTML, CSS, and JavaScript, guarantees a positive user experience. In order to demonstrate Voxbot Assistant's possible uses in customer service, personal help, and educational tools, this paper examines the architecture, development process, and features of the tool. The project shows how web technology and AI may be combined to create an interactive and flexible. The principal aim of Voxbot Assistant is to function as an adaptable and intuitive interface for information retrieval, suggestion generation, and conversation. With its user-friendly UI and AI-powered powers, Voxbot Assistant seeks to improve user experience, expedite communication, and offer helpful support in a range of areas. This research aims to investigate how artificial intelligence (AI) chatbots might improve human-computer interaction, automate repetitive jobs, and offer individualized support. Voxbot Assistant is a significant advancement in the development of AI-driven conversational interfaces, providing users with a potent tool for information retrieval and meaningful interactions by utilizing cutting-edge technology like speech recognition and natural language processing.

II. Motivation

1. Rise in AI Adoption: As AI is used more and more in many industries, people are becoming more interested in using AI-powered solutions to automate tasks and boost productivity. One common usage of AI is chatbots, which provide consumers with a convenient means of interacting with systems and retrieving information.

2. User Convenience: In order to complete activities or receive information, users using traditional interfaces frequently have to go through intricate menus or forms. By providing a more conversational and natural interface, chatbots improve customer satisfaction and convenience by enabling users to speak in simple terms and get prompt responses.

3. 24/7 Availability: Chatbots may work around the clock without the need for breaks or relaxation, giving users round-the-clock help

and support. This sets them apart from human agents. This constant accessibility is especially helpful in situations like customer service, when quick replies are crucial.

4.Contextualization and Personalization: Chatbots can provide customized recommendations and responses by using natural language processing techniques to comprehend user intents and preferences. The capacity to contextualize contacts improves the communication's efficacy and relevance.

5.Possibilities for Education and Learning: Working on a chatbot project such as Voxbot Assistant offers excellent chances for education in artificial intelligence, natural language processing, web development, and human-computer interaction. It enables developers to investigate cutting-edge technologies and use them in realistic, useful contexts.

III. Methodology

3.1 An analysis of requirements:

The requirement analysis phase comprises performing market research to identify current chatbot solutions, evaluating their advantages and disadvantages, and identifying chances for difference, in addition to determining the target audience and comprehending user demands. This involves examining chatbots that rival the Voxbot Assistant, spotting functional or user experience deficiencies, and developing the Voxbot Assistant's unique selling proposition.

3.2 Technological Opposition:

Technology selection includes assessing cloud-based services and platforms for hosting and deploying the chatbot in addition to choosing programming languages, frameworks, and libraries. This entails picking a reputable hosting company, deciding on the best deployment strategy (on-premises, cloud-based), and guaranteeing the infrastructure's scalability and dependability to handle any future spikes in user traffic.

3.3 Development of Backend:

Backend development involves not only developing fundamental functionalities like NLU, answer generation, and dialog management, but also integrating capabilities for data storage, user authentication, and integration with third-party services or APIs. This entails putting in place authentication methods like OAuth, guaranteeing data security and privacy compliance, and interacting with outside services to get pertinent data or carry out tasks on the user's behalf.

3.4 Design for the front end:

Frontend design includes not only making the interface clear and easy to use, but also making sure the chatbot works well across a range of screens and devices. This entails putting responsive design principles into practice, making mobile device performance optimal, and guaranteeing accessibility for people with disabilities.

3.5 User Input and Revisions:

User feedback and iteration include gathering feedback via user testing as well as evaluating user interaction data to find trends, preferences, and areas that need improvement. This entails utilizing sentiment analysis to gauge customer happiness, analytics tools to monitor user engagement, and A/B testing to gauge how well various features or design modifications work.

3.6 Implementation and Upkeep:

Deployment and maintenance also include setting up automated update and version control methods to make sure the chatbot always has the newest features and security patches. This is in addition to regular monitoring and maintenance. To expedite the deployment and maintenance process, this entails putting in place continuous integration/continuous deployment (CI/CD) pipelines, version control systems like Git, and automated testing frameworks.

3.7 Flow Chart:

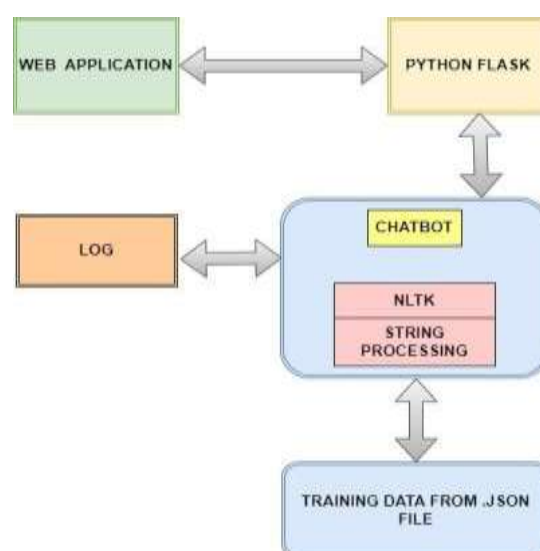


Figure 1:Flow Chart

III. Literature Review

4.Overview of Conversational Agents:

Chatbots have become increasingly prevalent across various industries, driven by advancements in artificial intelligence (AI) and natural language processing (NLP). These automated conversational agents can simulate human-like interactions, providing users with information, support, and entertainment. This literature review explores the evolution, technologies, applications, and challenges associated with chatbots, with a focus on the development of Voxbot, a chatbot designed to assist users with their inquiries.

4.1 Chatbot Evolution:

The idea of chatbots originated in the 1960s when Joseph Weizenbaum developed ELIZA, a pioneering computer tool for natural language processing. ELIZA demonstrated the possibility for human-computer interaction by simulating speech using pattern matching and replacement methods. Since then, chatbots have developed dramatically, with important innovations like Jabberwacky and ALICE, and Siri marking key milestones in the development of conversational agents.

4.3 Frameworks and Technologies:

Contemporary chatbot programming makes use of several frameworks and technologies. Natural language generation (NLG) and natural language understanding (NLU) are essential elements that allow chatbots to interpret user input and produce relevant responses. A number of NLP frameworks and libraries, including spaCy, TensorFlow, Natural Language Toolkit (NLTK), and Dialogflow, offer powerful capabilities for creating complex chatbots.

4.4 Processing of Natural Language (NLP):

NLTK: An effective library that provides tools for text processing, tokenization, stemming, tagging, parsing, and other tasks when working with human language data.

spaCy :It is an industrial-grade natural language processing (NLP) toolkit that supports tasks such as dependency parsing, named entity recognition, tokenization, and part-of-speech tagging. It is designed for both ease of use and performance.

TensorFlow: A JavaScript model building and training framework that is open-source and can be integrated with web-based chatbots. It contains TensorFlow.js.

4.5 Dialog Management:

Rasa: An open-source framework with capabilities for entity extraction, custom action handling, and intent recognition that is used to construct conversational artificial intelligence.

Dialogflow: A Google-owned platform that builds conversational user interfaces for mobile apps, messaging services, and webpages using natural language processing (NLP).

4.6 Difficulties in Developing Chatbots:

Overcoming multiple obstacles is necessary to develop efficient chatbots. Natural Language interpreting (NLU) is complicated by variations in language use, slang, and context, making it difficult to achieve high accuracy in interpreting various and complex user inputs. Sustaining the context of discussions across several exchanges is crucial for offering pertinent and cohesive answers, underscoring the need of proficient context handling. It can be challenging to create conversational experiences for chatbots that are natural, fluid, and human-like in order to keep consumers interested and satisfied. Furthermore, reliable API development and effective data processing are needed for chatbots to be easily integrated with databases and current systems in order to deliver correct and timely information.

IV. Methodology

A systematic approach is used in the development of the Voxbot Assistant chatbot to guarantee the production of a reliable, approachable, and efficient conversational agent. The approach is broken down into multiple crucial stages:

- 1.Understanding user needs, defining the project scope, and identifying the target audience are the goals of requirement analysis. To gather user needs, surveys and interviews are conducted. Existing chatbots are analyzed to find gaps, and the functionality, features, and interactions required from the chatbot are defined.
- 2.Selecting the right technologies to deploy the chatbot is the goal of the technology selection process. This entails assessing various programming languages, frameworks, and libraries; choosing tools for front-end design, natural language processing (NLP), and back-end development; and completing the tech stack, which includes Flask and Python for HTML and backend development.
- 3.Backend development is the process of creating the chatbot's essential features. This entails building response generation and dialog management modules, implementing natural language understanding (NLU) using NLP libraries like NLTK, spaCy, or TensorFlow, and establishing APIs to process user input and provide relevant responses.
- 4.Frontend design, with the goal of developing a simple and easy-to-use interface for communicating with the chatbot. This include creating input forms and UI elements to promote user interaction; designing web pages and user interfaces with HTML, CSS, and JavaScript and making sure that the design is responsive for the best possible user experience on a variety of devices.
- 5.The goal of integration and testing is to make sure the chatbot performs as intended by connecting the frontend and backend components. In order to do this, the frontend interface and backend APIs must be integrated. Unit, integration, and end-to-end testing must also be carried out, and any problems, defects, or performance or functionality discrepancies must be found and fixed.
- 6.To make sure the chatbot works as intended, integration and testing are used to combine frontend and backend components. This include unit testing, integration testing, end-to-end testing, and integrating the backend APIs with the frontend interface. It also include finding and resolving faults, defects, and performance or functionality anomalies.
- 7.User Feedback and Iteration: Based on user feedback, the chatbot's usefulness and efficacy are assessed. In order to improve the chatbot's performance and fix any usability issues, this entails launching the chatbot for user testing, gathering and evaluating user feedback about usability, performance, and satisfaction, and putting iterative improvements into practice based on feedback.

VI.Implementation

The first step of the project is to set up a Flask environment, a Python web framework that was selected for its adaptability and simplicity. Flask is a great option for creating a chatbot interface since it makes it easier to create web applications and APIs. Make sure Flask is installed within the Python environment before configuring the Flask environment. Using pip, a Python package management system, this can be done by running the pip install Flask command. A new Flask application is started by generating a Flask app object after Flask has been installed.Setting up the Flask application to serve static files and HTML templates is the next stage. User-facing web pages are structured by HTML templates, with the addition of static files like CSS and JavaScript to improve their look and feel. Flask facilitates straightforward asset organization by allowing developers to choose directories for the storage of various templates and files .After setting up the Flask environment, the project moves on to build the required HTML templates. The design and content of the webpages that consumers will interact with are specified by these templates. Templates like welcome.html and chat.html customize and expand the basic template to fit particular application pages .Users land on the welcome.html template, which provides a quick overview of the chatbot and walks them through the first few steps of interaction. Welcome messages, chat interface usage instructions, and connections to useful resources or help channels could all be found on this page.The primary chat interface, on the other hand, through which users interact with the chatbot, is represented by the chat.html template. Typically, this design has an input field where users may type messages, a chat history display section, and interactive message sending and receiving elements. The user's browser and the Flask backend can communicate in real-time through JavaScript code placed in the template, which makes it easier for the user and the chatbot to communicate.

1. Autocomplete Functionality :Your jQuery and AJAX autocomplete feature improves user engagement by presenting suggestions in real time depending on user input. Here's how to describe its application and advantages:

Implementation: By combining the Autocomplete widget from jQuery UI with the routing and AJAX (/autocomplete endpoint) features of Flask, you've created a system in which users type and the server replies by fetching suitable suggestions from a database or predetermined list as they type.

Advantages:

Improved User Experience: Instant ideas are given to users, which expedites their communication with the chatbot.

Decreased Errors: By providing users with appropriate selections, it helps avoid typos and mismatches.

Scalability: As your application develops, it may be readily expanded to accommodate bigger datasets or more intricate recommendation algorithms.

2.Reactions :We have set up a replies dictionary or something similar to effectively handle particular user inputs. For a closer look, consider this:

Structure: The replies dictionary associates instructions or user inquiries with the appropriate actions or responses from the bot.

Advantages:

Efficiency: Using preset triggers, quickly adjusts the flow of the conversation.

Customization: Responses are simple to update and alter without affecting the underlying logic.

Personalization: May increase engagement by adjusting responses according to user context or previous session information.

3.Restaurant Listings :There is a /restaurants route in your application that provides information on restaurants, including their menu, rating, and cuisine. You can elucidate on its functionality as follows:

Details Display: Gives consumers the most important details about eateries so they can make educated choices or do more research.

Possible Improvements:

Allow consumers to filter based on cuisine, rating, or location with interactive features.

Integration: Provide a link to outside services or APIs to get real-time restaurant data updates.

User Reviews: To improve listings, provide user-generated information such as reviews and ratings.

4.CSS and HTML: The chatbot and suggestions container have a clear, legible interface thanks to the frontend design's use of structured HTML and CSS. Let's take a closer look:

Structure: Organized HTML elements with semantic tags enhance accessibility and SEO.

Styling: CSS rules provide consistent branding and layout, optimizing usability.

Responsiveness: Design adapts well across different screen sizes, ensuring a seamless experience on desktop and mobile devices.

5.Ideas and Subjects: To efficiently direct user interactions, you've incorporated suggested subjects and specified themes into the UI. Here's why this is advantageous:

Navigation Aid: Assists users in navigating through different features or FAQs without having to formulate specific questions.

Contextual Guidance: Makes subject suggestions depending on user input or current talks.

User engagement: Promotes interaction and discovery, which raises user retention and overall satisfaction.

5.Functionality Recommendations :Adding more features to your chatbot's UI will improve user experience and application utility even more:

Create features for in-depth inquiries about particular establishments, such as menu items, operating hours, or exclusive deals.

Food Ordering Backend: If necessary, integrate with restaurant APIs or form alliances to ensure smooth transactions. Implement backend logic for food ordering.

VII . RESULTS

Output window: The Dash application locally. After executing the code, we'll create a temporary public URL that the programmer may view.

```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 110-705-864
```

Figure 2: Output Window

Style of Web Application : The "Food Seeker AI Chat" web application, which is made with Flask and Python, has a modern, minimalistic design with images related to food. This ensures that the application is usable on a variety of devices and keeps the branding constant.



Figure 3 : Style of Web Application

Interacting With VoxBot Assistant: With its easily accessible and comprehensive recipe database, the VoxBot Assistant helps customers in their culinary undertakings in an efficient manner. Potential future developments could concentrate on growing recipe libraries and refining tailored recommendations.

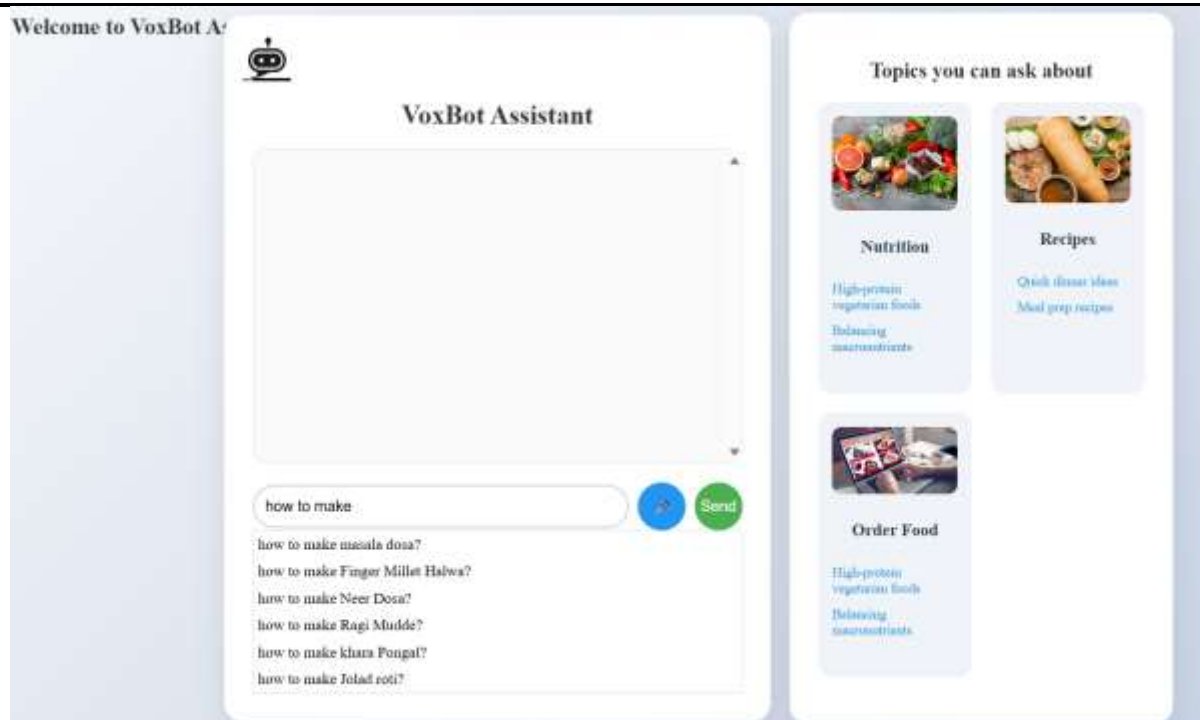


Figure 4: Querying the bot

Bot Output : Clickable links are used by the VoxBot Assistant to deliver recipe details. By clicking on these links, visitors are taken to specific sites that provide comprehensive ingredient lists and cooking directions, providing a seamless method of accessing comprehensive recipe information.

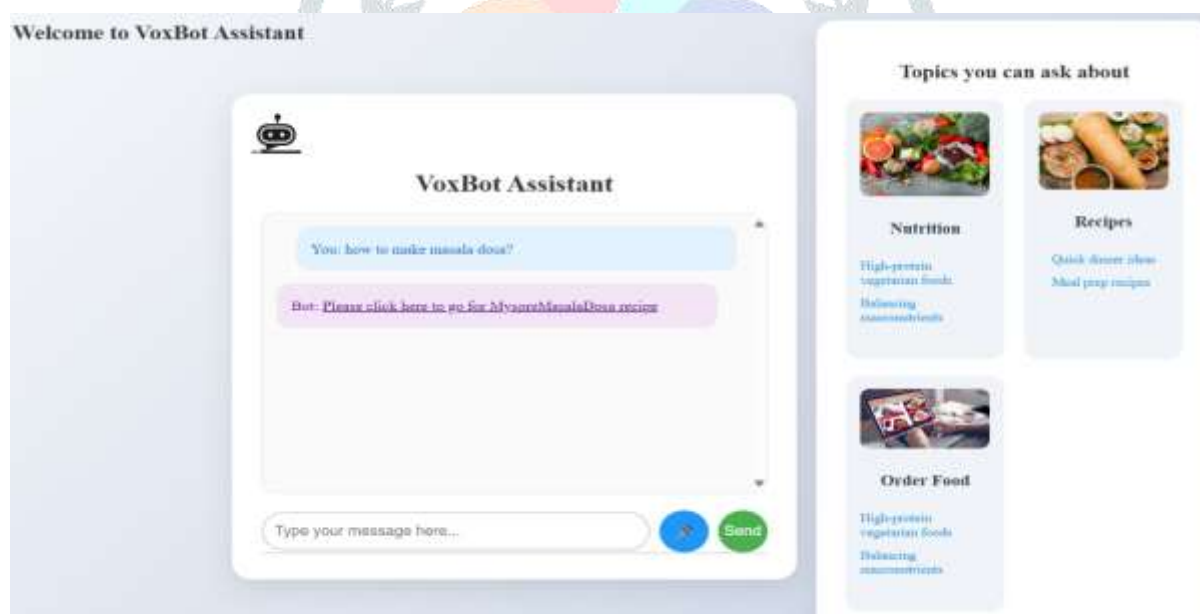


Figure 5: Bot Output

Post Click Instructions: Users are taken to a thorough page with extensive recipe instructions after clicking the offered link. This page contains extra cooking suggestions, a list of ingredients, and a step-by-step recipe for making the dish. Because of the format's user-friendly design, following and executing the recipe is made simple.

Ingredients

- Dosa rice
- Urad dal
- Toor dal
- Chana dal
- Fenugreek seeds
- Potatoes
- Onion
- Green chilli
- Ginger
- Dried red chillies
- Turmeric powder
- Mustard seeds
- Cumin seeds
- Butter
- Salt

Mysore Masala Dosa Recipe

1. Soak rice, urad dal, toor dal, chana dal, and methi seeds for 4-5 hours. Grind to a smooth batter, cover, and let it ferment overnight. Add washed pooha to the batter before grinding.
2. To make the red chutney, heat oil in a pan and add chana dal, ginger, garlic, and onion. Sauté until the onion is soft. Add dried red chillies, turmeric, salt, and a little water. Blend the mixture to a smooth paste.
3. For the potato filling, heat oil in a pan and add mustard seeds. Let them splutter. Add cumin seeds, chana dal, urad dal, hing, dried red chili, and curry leaves. Sauté until the dals are golden.
4. Add finely chopped onion, green chili, and ginger. Sauté until the onions are golden brown.
5. Add boiled and mashed potatoes, turmeric powder, salt, and a little water. Mix well and cook for 5 minutes. Garnish with chopped coriander leaves and lemon juice.
6. Heat a non-stick pan and pour a ladle of the dosa batter in the center, spreading it in a circular motion to make a thin dosa. Drizzle some butter around the edges.
7. Spread a spoonful of the red chutney evenly on the dosa. Place a portion of the potato filling in the center, fold the dosa over, and cook until crispy.
8. Serve hot with coconut chutney and sambar.



Figure 6: Post Click instructions

Ordering food with using VoxBot Assitant: Apart from offering recipe details, the "Food Seeker AI Chat" bot is being created to make ordering food easier. Through the chat interface, users may engage with the bot to explore menus of nearby restaurants, choose items, and place orders. The purpose of this feature is to improve user convenience and efficiency by streamlining the meal ordering process.

Welcome to VoxBot Assistant


VoxBot Assistant

You: I want to order something

Bot: [Click here to view restaurants and order food](#)


Type your message here...

Topics you can ask about




Nutrition

High-protein
vegetarian foods
Balancing
macronutrients



Recipes

Quick dinner ideas
Meal prep recipes




Order Food

High-protein
vegetarian foods
Balancing
macronutrients

Figure 7: Ordering Food

FoodExpress

Search for restaurants or cuisines...




Tasty Bites

Italian

★ 4.5

Downtown • 2.2 km away

Order Now




Sushi Paradise

Japanese

★ 4.7

Uptown • 3.1 km away

Order Now




Burger Haven

American

★ 4.2

Midtown • 1.8 km away

Order Now



Spice Route

Indian

★ 4.6

West End • 4.0 km away

Order Now

Figure 8: Output of the Query

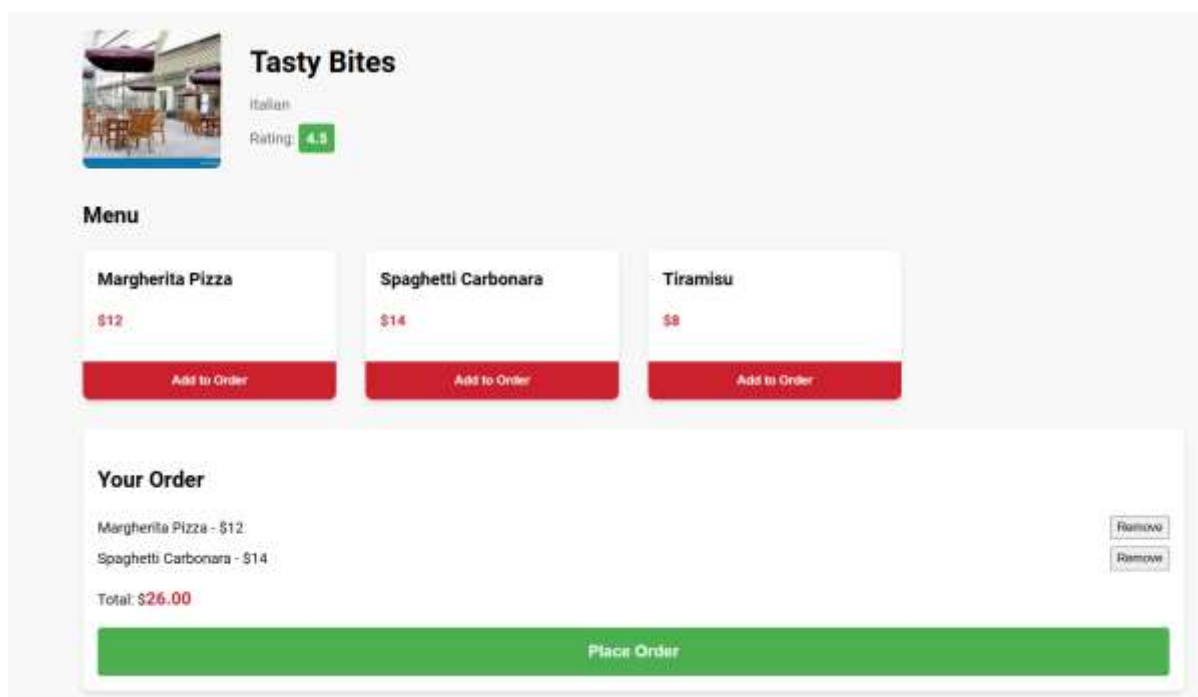


Figure 9: Placing an order



Figure 10: Message from the Bot

VII.CONCLUSION AND FUTURE PLAN

A conversational assistant for recipe queries can be created with Flask and Python, as the "Food Seeker AI Chat" project has effectively shown. Cooking is made easier for consumers by the bot, which effectively gives them ingredient lists and comprehensive recipe instructions.

In the future, we want to add more recipes to the bot's repertoire and provide personalized recommendations that are tailored to the tastes of the user. Furthermore, our goal is to enhance the bot's comprehension of natural language in order to provide even more precise and beneficial responses.

IX.REFERENCES

- [1] K. Bala ,M. Kumar, S.Hulawale, and S. Pandita, "Chat-Bot ForCollege Management System Using A.I", International ResearchJournal of Engineering and Technology (IRJET), Vol. 04, Issue.11, pp.2395-0072, 2017
- [2] Thomas, N. T. (2016). An e-business chatbot using AIML and LSA. 2016 International Conference on Advances in Computing, Communi-cations and Informatics (ICACCI). doi:10.1109/icacci.2016.7732476
- [3] N. Thomas, " An e-business chatbot using aiml and lsa," in Advances in Computing, Communications and Informatics (ICACCI), 2016 Interna-tional Conference on. IEEE, 2016, pp. 2740–2742.
- [4] S. Reshmi and K. Balakrishnan, "Implementation of an inquisitive chatbot for database supported knowledge bases," Sadhan a , vol. 41, no. 10, pp. 1173–1178, 2016
- [5] Prof.K.Bala, Mukesh Kumar, SayaliHulawale, SahilPandita,"Chat-Bot For College Management System Using A.I" International Research Journal of Engineering and Technology (IRJET) Volume: 04, Issue: 11, Page no: 2030-2033— Nov 2017.
- [6] Lancaster Stemmer Algorithm, Available at <https://www.nltk.org/api/nltk.stem.lancaster.html>
- [7] Porter Stemmer Algorithm, Available at the link <https://tartarus.org/martin/PorterStemmer> [8] Amey Tiwari, Rahul Talekar, Prof.S.M.Patil, "College Information Chat Bot System" International Journal of Engineering Research and General Science (IJERGS) Volume: 5, Issue: 2, Page no: 131-137— MarchApril.