An E-commerce Sales Chatbot

Aneesh Adithya SR

Abstract—Machine-This project presents a rule-based shopping assistant designed to enhance product discovery on e-commerce platforms through simple text-based interactions. Instead of using complex artificial intelligence or machine learning algorithms, the system relies on keyword-based matching logic to respond to user queries. Built using the Flask web framework, the assistant provides a clean and responsive user interface developed with Bootstrap. The backend connects to a MySQL database that stores product information, user credentials, and chat history. Users can search for products based on specific attributes such as name, brand, color, or price, and the chatbot returns relevant matches based on predefined conditional logic. A secure login system is implemented using password hashing and session management to protect user data and maintain chat continuity. The assistant offers an intuitive and user-friendly experience, demonstrating how basic rule-based logic can still provide effective search functionality. Future enhancements could include AI integration, recommendation systems, inventory tracking, or automated customer support for more advanced capabilities.

Index Terms—Shopping Assistant, Conversational AI, E-commerce Chatbot, Product Search, Natural Language Processing, MySQL Database, User Authentication, Flask Framework, Bootstrap Interface, Personalized Recommendations, Retail Technology, Chat History Persistence, Responsive Design, Secure Login, Session Management.

Introduction

With the growing popularity of e-commerce platforms, users often face the challenge of searching through vast product catalogs to find what they need. To address this, we developed a rule-based shopping assistant that allows users to interact with the system using natural language inputs. Instead of using complex AI or machine learning techniques, our system relies on keyword matching logic—implemented through conditional statements—to identify relevant products based on user queries. The assistant is built using the Flask web framework for backend logic, MySQL for storing user and product data, and Bootstrap for creating a responsive and user-friendly front-end interface. The system supports user authentication with password hashing for secure login and manages user sessions to retain chat context. Users can search for products based on attributes like name, brand, color, or price, and receive appropriate matches from the database. This project demonstrates how even simple logic-based approaches can enhance user interaction in e-commerce applications, offering a functional and intuitive product search experience without relying on complex AI algorithms.

Technology Stack Used

The project utilizes a combination of front-end, back-end, and database technologies to build a functional and user-friendly shopping assistant. For the front-end, HTML, CSS, and Bootstrap were used to create a responsive and visually appealing user interface. JavaScript is used for dynamic page behavior and enhancing user interactions. On the server side, the application is built using Flask, a lightweight Python web framework that handles routing, session management, and server-side logic. The back-end logic includes condition-based keyword matching to respond to user queries without the use of AI or machine learning. MySQL serves as the relational database, storing essential information such as user credentials, product details, and chat history. Password hashing ensures secure login functionality, while Flask sessions help maintain stateful user interactions. Overall, this stack provides a simple yet effective environment for developing a rule-based conversational interface suitable for e-commerce platforms.

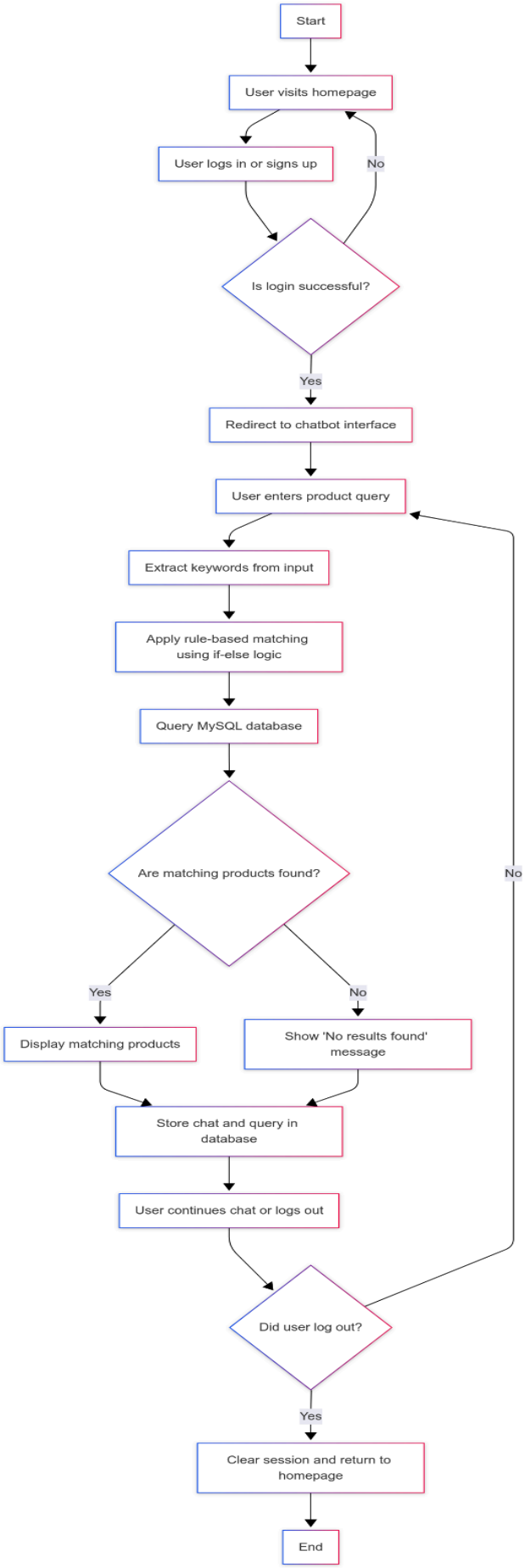
.

Methodology

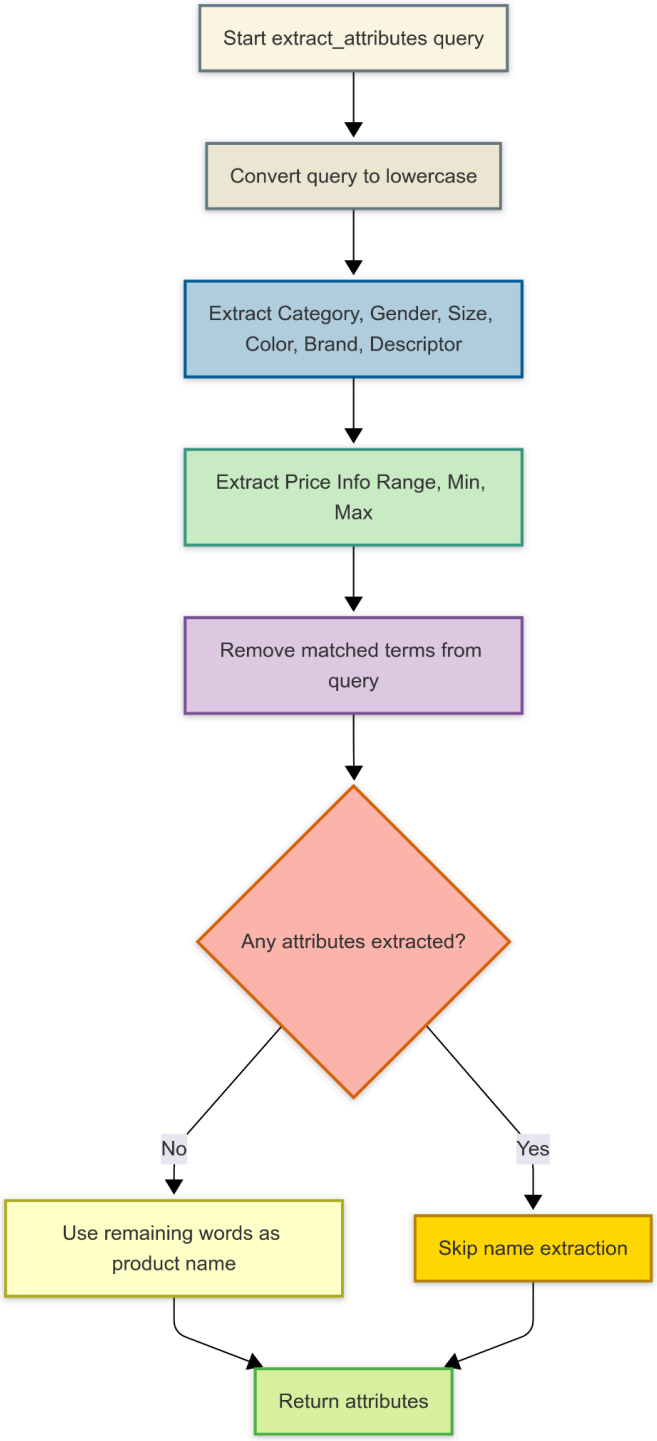
The development of the shopping assistant was carried out through a structured, step-by-step approach to ensure clarity, functionality, and security. First, the requirements were gathered and analyzed, focusing on enabling users to search for products using simple text inputs. Next, the database schema was designed using MySQL to store user information, product details, and chat logs. After that, the Flask web framework was set up to handle server-side logic, including user authentication, routing, and form submissions.

The frontend interface was developed using HTML, CSS, and Bootstrap to provide a responsive and intuitive user experience. JavaScript was used to improve interactivity. A rule-based matching algorithm was implemented using if-else conditions to process user inputs and return relevant products based on keywords such as name, brand, color, and price.

To ensure secure access, password hashing was applied for user credentials, and Flask sessions were used for managing login states and retaining chat history. Finally, the system was tested manually to verify functionality, accuracy of search results, and ease of use across different user scenarios.

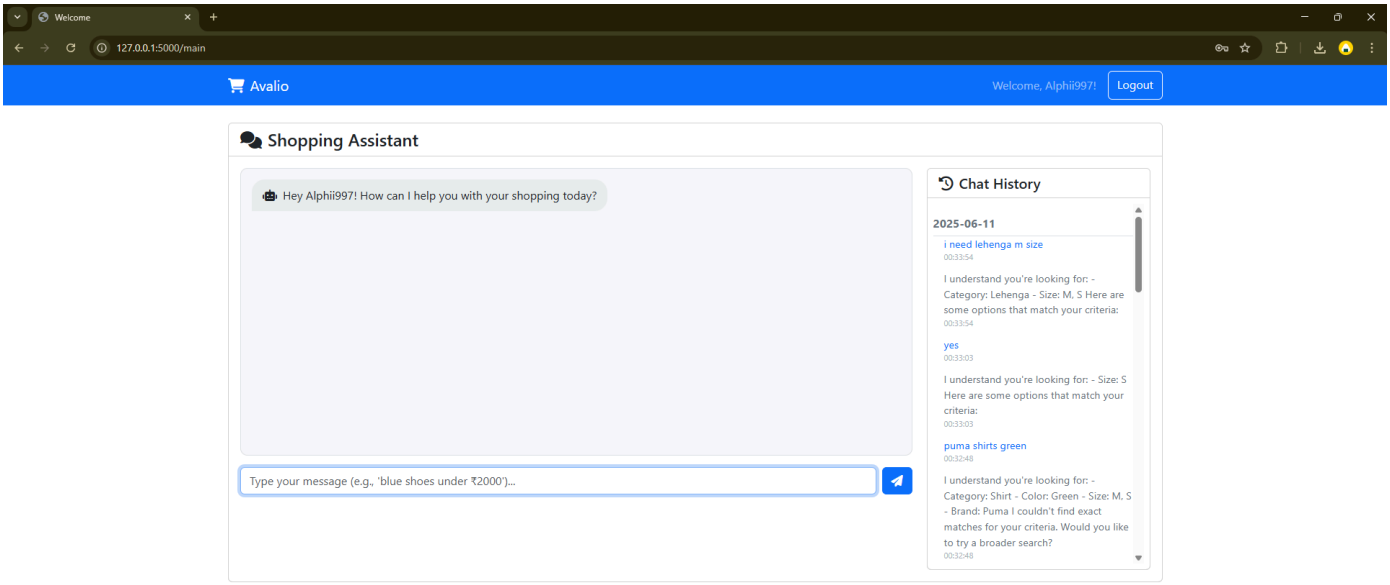


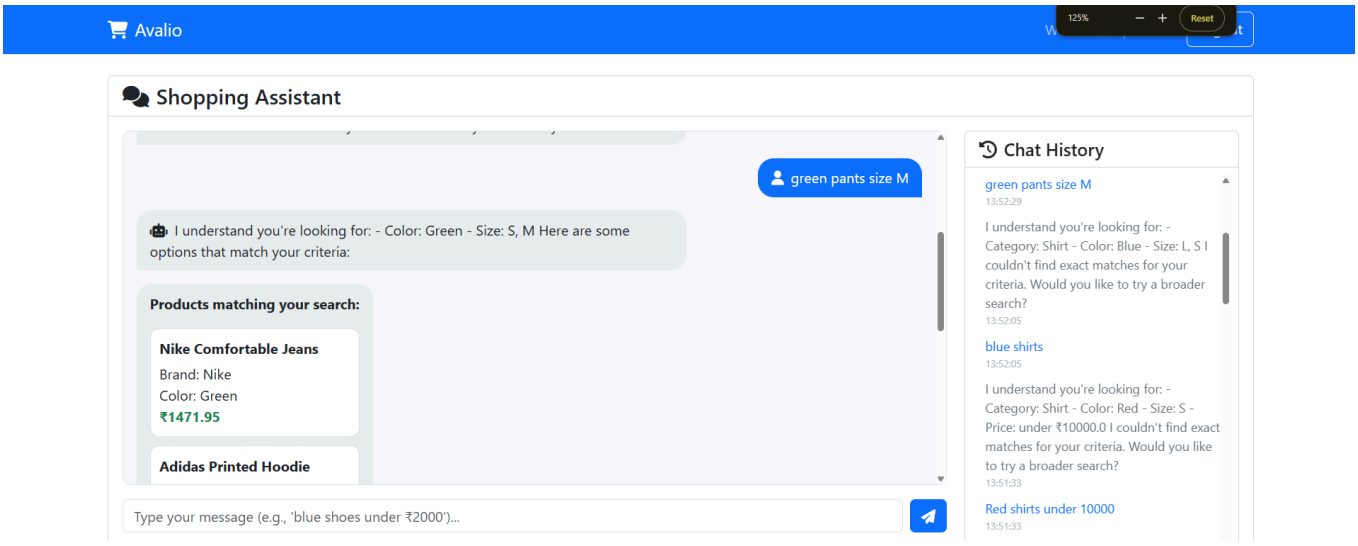
Flowchart-Methodology

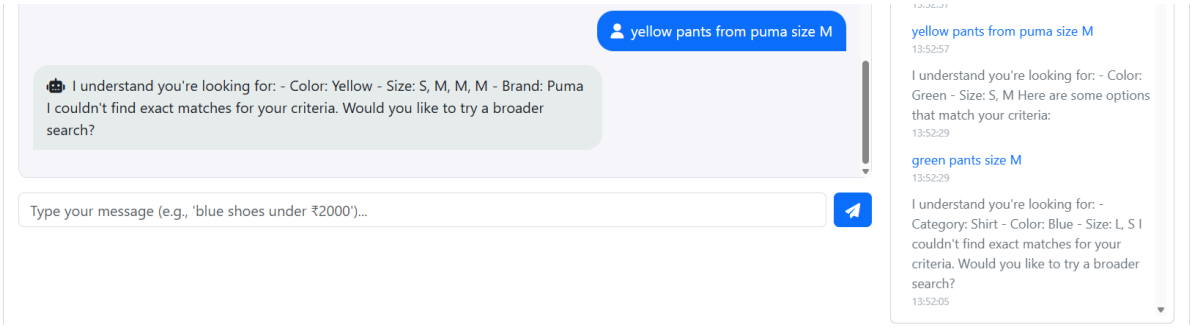


Attribute Extraction Logic Flow

IV. Sample Queries







When there are no results in Database

Results

The rule-based shopping assistant was successfully implemented and tested across multiple user queries simulating realistic product searches. The system accurately extracted relevant attributes such as product category, brand, color, size, gender, and price range from natural language queries using predefined keyword lists and conditional logic. When users entered search phrases like "show me red Nike t-shirts under 1000" or "women's black jeans from Levis," the chatbot consistently matched these inputs with corresponding products in the MySQL database and returned appropriate results.

All chat interactions during an active session were stored in real time, enabling persistence and personalization. The login system was validated to ensure secure access, and password hashing worked correctly to protect user data. The Bootstrap interface provided a responsive and user-friendly experience across devices.

During testing, the chatbot handled both specific and vague queries effectively, demonstrating robustness in attribute extraction and matching. While the system does not use AI, it achieved a high level of accuracy in product retrieval within the scope of predefined conditions.

Problems Faced

One of the main challenges in this project was handling natural language queries without using any AI or machine learning models. Extracting meaningful attributes such as product names, colors, brands, sizes, and price ranges required writing extensive rule-based logic using conditional statements and regular expressions. This approach quickly became complex and harder to maintain as more variations in user phrasing were introduced.

Unlike AI-based systems that can generalize and learn from examples, a rule-based approach lacks flexibility. For instance, minor changes in user input format could lead to failed matches unless explicitly handled in the code. Additionally, accounting for spelling errors, synonyms, or complex sentence structures would require disproportionately more lines of code and manual effort.

Another difficulty was ensuring consistent database matching and real-time chat storage without introducing performance bottlenecks. Designing a scalable and maintainable system architecture while keeping it simple and functional was a constant balancing act throughout development.

We also faced limitations in sourcing a relevant dataset. Since no existing dataset matched our project's specific requirements for product categories and attributes, we manually created a synthetic dataset consisting of 100 rows. This involved carefully crafting product entries with diverse attributes to support robust rule-based query handling and testing.

Future Scope

- Integration of NLP or AI Models: Replacing the rule-based query parser with a machine learning or NLP model (e.g., spaCy, BERT) to better understand user intent, handle varied phrasing, and support natural conversations.
- Recommendation System: Adding a product recommendation engine that uses user preferences, past interactions, or collaborative filtering to suggest products intelligently.
- Spelling Correction & Synonym Handling: Implementing fuzzy matching, spelling correction, and synonym support to make the system more robust to human input errors and varied vocabulary.
- Admin Dashboard & Inventory Management: Building an admin panel for adding, updating, or removing products dynamically, along with stock tracking and analytics.

Conclusion

This project successfully demonstrates the development of a functional shopping assistant chatbot using a rule-based approach without relying on artificial intelligence. By combining a Flask web backend, a Bootstrap-based frontend, and a MySQL database, the system enables users to search for products using natural language queries. Despite the absence of AI, the chatbot was able to extract relevant attributes such as product category, brand, color, and price range through well-defined logic and keyword matching. The integration of a secure login system and real-time chat storage enhances usability and personalization.

While the limitations of rule-based methods were evident—especially in handling complex or ambiguous queries—the project highlights the feasibility of building a usable and responsive conversational interface with limited resources. Creating a synthetic dataset also enabled effective testing and demonstration of the system. This work lays a solid foundation for future enhancements, including AI integration and intelligent recommendation features to improve scalability and user experience.

References

M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed., Sebastopol, CA: O'Reilly Media, 2018. [Online]. Available: https://flask.palletsprojects.com/

Oracle Corporation, MySQL 8.0 Reference Manual, [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/