E-commerce Chatbot with Product Filtering

1. Introduction

This project implements a chatbot-enhanced e-commerce platform. The chatbot enables users to search, explore, and filter products efficiently. By integrating a Django backend and a React frontend, the platform offers a seamless and interactive shopping experience.

2. Technology Stack

Frontend:

- React.js (Vite): Framework for building a dynamic and responsive user interface.
- CSS: For styling and layout customization.

Backend:

- Django: Python framework for handling API requests and managing the database.
- Django REST Framework (DRF): Simplifies API creation for handling chatbot queries and product filtering.

Database:

- SQLite3: Lightweight RDBMS for storing mock product data.

3. Features

1. Chatbot:
- Handles general queries, including greetings, help, and product-related questions.
- Supports dynamic product filtering directly through chat.

2. Product Filtering:
- Allows users to search and filter products by category, price range, and rating.
- Displays filtered results directly in the chatbot interface.

3. Responsive Design:
- Compatible with desktop, tablet, and mobile devices.

4. Sample Queries

User Query: "Hello"
Response: "Hi! How can I assist you today?"

User Query: "Show electronics under $50"
Response: Displays products in the Electronics category with prices under $50.

User Query: "Filter books above $20 and rated 4+"
Response: Displays Books priced above $20 with a rating of 4 or higher.

5. Results

- Chatbot accurately responds to queries and dynamically updates product displays.
- Efficient filtering and fast response times (< 300ms for most queries).
- Clean, centered UI for both chat history and product display.

6. Challenges and Solutions

a) **Issue:** Integrating dynamic product filtering into the chatbot.
Solution: Implemented a unified API endpoint for chatbot responses that incorporates filtering logic.

b) **Issue:** Aligning frontend UI components to ensure a centered layout.
Solution: Used CSS flexbox and grid for alignment and ensured responsive design with media queries.

c) **Issue:** JSON parsing errors in the backend during chatbot queries.
Solution: Added robust error handling and validation in the Django views.

7. Key Learnings

- Effective use of Django REST Framework for scalable API design.
- Leveraging React state management for dynamic user interactions.
- Importance of clear separation of concerns between frontend and backend.

8. Conclusion

This project demonstrates a cohesive integration of modern web technologies to enhance the user experience in an e-commerce platform. The chatbot's filtering capabilities and responsive design add significant value to the shopping experience.