E-commerce Chatbot

Abstract

In the competitive world of e-commerce, user engagement and ease of product discovery are critical. This project aims to develop a sales chatbot that enhances user interaction and product exploration through an intelligent, interactive interface. By combining React.js for the frontend, Flask for the backend, and SQLite for data management, the chatbot delivers an efficient, responsive shopping assistant that processes natural language and applies real-time filters to showcase products.

Objective

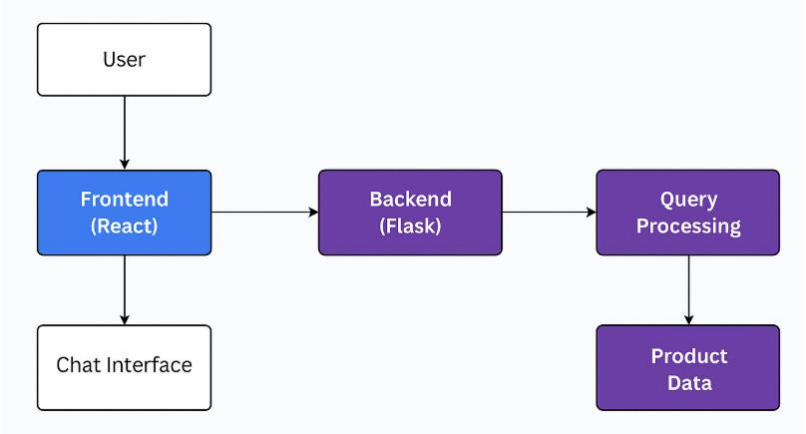
To design and implement a user-friendly, intelligent sales chatbot for an e-commerce platform, enabling users to search and explore products (like phones, headphones, and watches) easily. The chatbot simulates a real-time assistant capable of:

- Understanding search queries
- Displaying product details with images

Tools and Frameworks Used

Category	Tools/Frameworks
Frontend	React.js, HTML5, CSS3
Backend	Python Flask
Database	SQLite
Deployment	Vercel (Frontend), Render (Backend)
Version Control	Git & GitHub
Design Tool	Eraser.io

Architecture Diagram

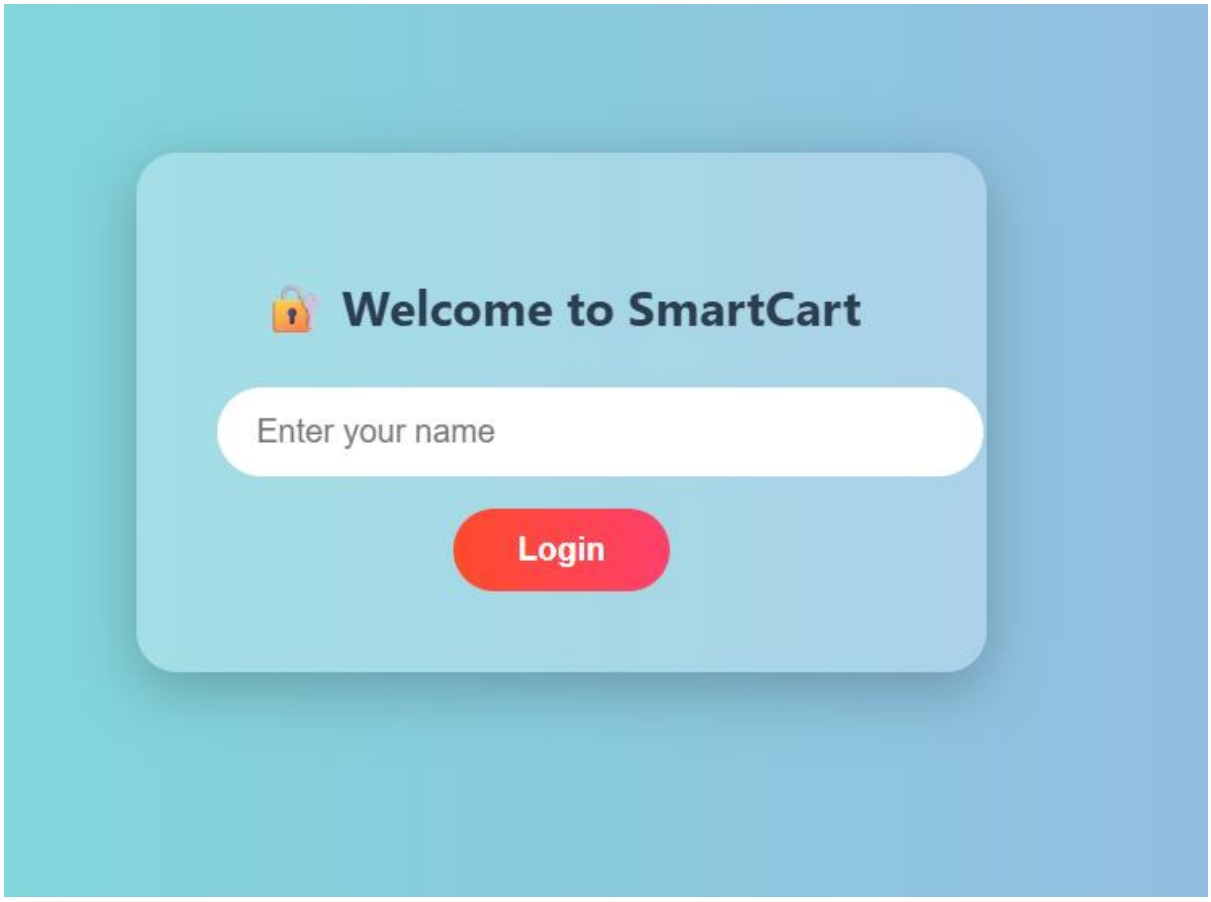


- **Frontend** handles user input, chat flow, login interface, and fetch calls.
- **Backend** processes chatbot logic and filters, queries the database.
- **Database** stores product inventory.

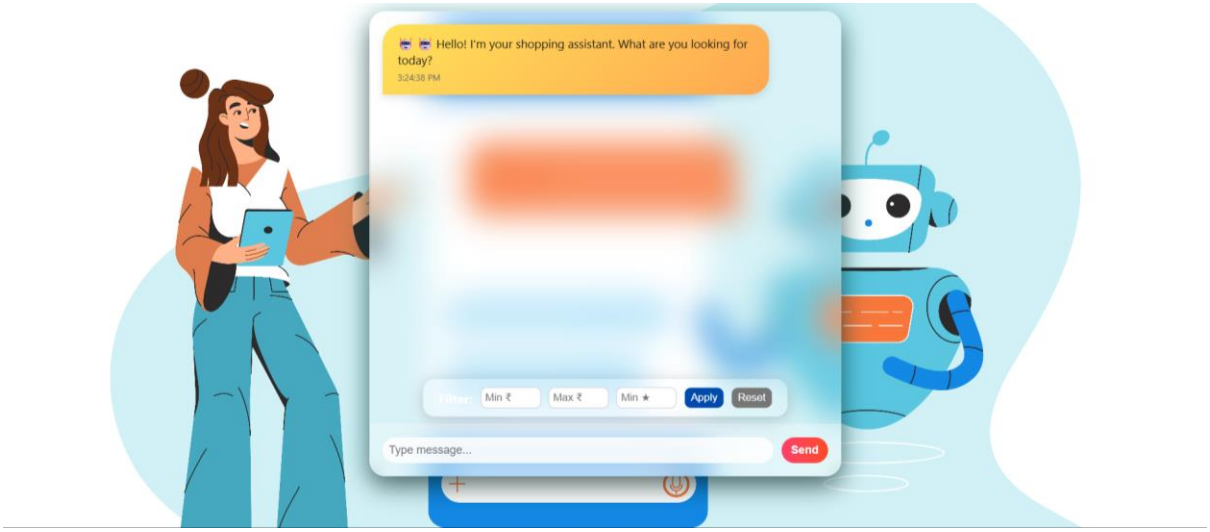
Kolupoti Sai Prakash

Screenshots

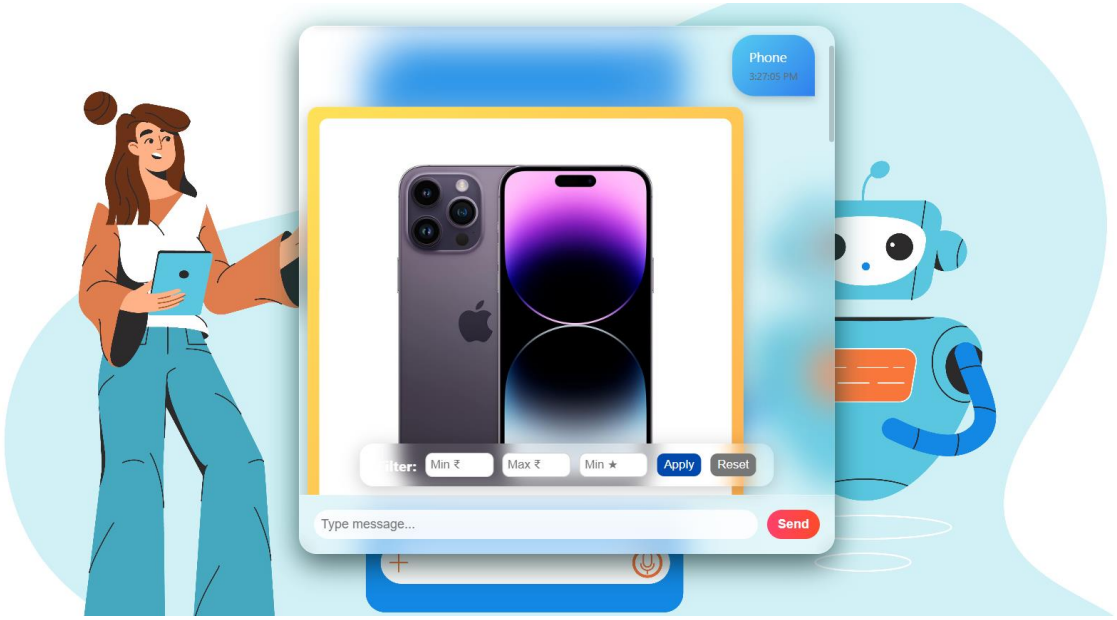
- Login Page

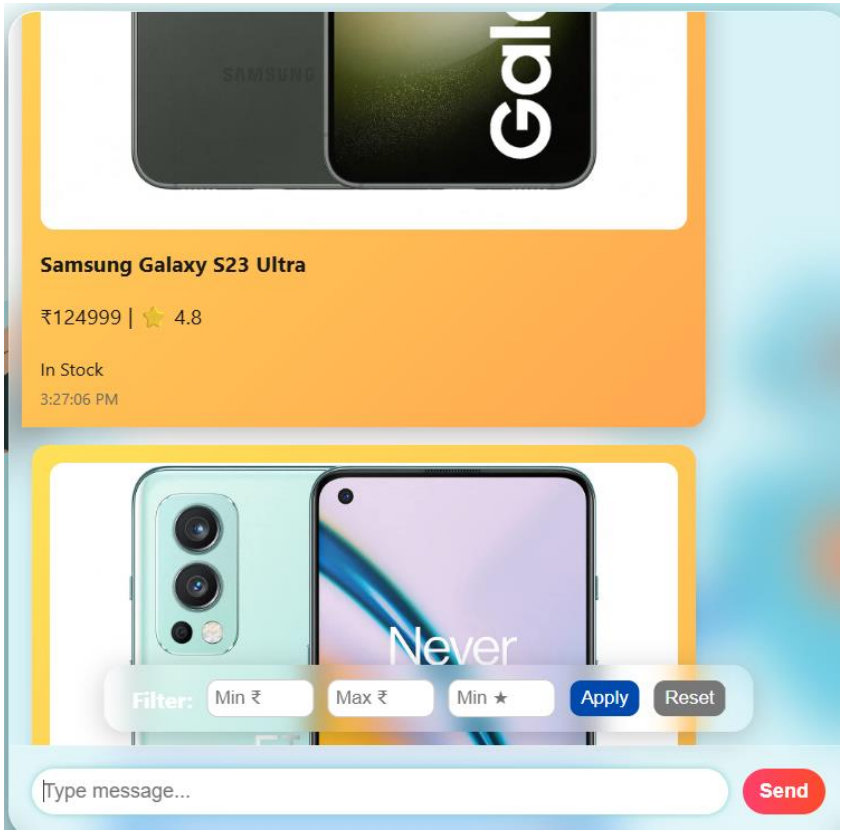


- Chatbot Interface



- Product Cards with Images, Price, Ratings





- Filter Bar (Min Price, Max Price, Min Rating)

Backend Flow & Database Schema

Flask API Endpoints

- `POST /chat` → Accepts user input, parses and returns product suggestions.
- `POST /filter` → Accepts filter params (`minPrice`, `maxPrice`, `minRating`) and returns matching products.

SQLite Schema

```
CREATE TABLE products (
id INTEGER PRIMARY KEY,
name TEXT,
category TEXT,
price REAL,
rating REAL,
availability TEXT,
image_url TEXT
);
```

- 20 mock product entries inserted manually with realistic data.

Challenges Faced & Solutions

Challenge	Solution
Handling natural user inputs	Used keyword parsing fallback messages
Styling across devices	Used responsive CSS grind and Flexbox layout
Deployment with both frontend and backend	Deployed React on Vercel, Flask on Render
Fetch calls not connecting due to CORS	Configured backend CORS settings with Flask-CORS
GitHub repo syncing	Managed branches and used `git pull --allow-unrelated-histories`

Conclusion

This chatbot project demonstrates how conversational UIs can enrich the user experience in e-commerce by combining simplicity with intelligent interactivity. From login to product exploration, users are guided through an efficient and visually engaging experience. The backend is lightweight yet powerful, using Flask and SQLite. The frontend is responsive and easy to extend. All these make it a complete demonstration of full-stack development with clear modularity and real-world use case simulation.

Deployment Links

- **Frontend (Vercel):** https://ecommerce-chatbot-gilt.vercel.app/login
- **Backend (Render):** https://ecommerce-chatbot-backend-jn1d.onrender.com
- **GitHub Repository:** https://github.com/saikolupoti/Ecommerce-Chatbot
- **ChatBot:** https://ecommerce-chatbot-gilt.vercel.app/login