Project Report: E-commerce Sales Chatbot

---

1. Introduction

In this project, we developed an **E-commerce Sales Chatbot** aimed at improving customer experience in an e-commerce platform. The chatbot allows users to interactively search for products based on categories like **electronics, books,** and **clothing**. The backend system processes product queries and returns relevant data to the chatbot interface, making it easy for users to explore and purchase items.

---

2. Technology Stack

The technology stack for this project was chosen to meet the requirements for building an interactive chatbot with a strong backend support system.

- **Frontend**:
  - **React.js**: A JavaScript library for building dynamic user interfaces.
  - **HTML5 & CSS**: To structure and style the web page.
  - **Axios**: For making HTTP requests to communicate with the backend.
- **Backend**:
  - **Flask (Python)**: A micro-framework used to build the RESTful API for handling product search and fetching results.
  - **SQLite (for mock database)**: A lightweight relational database to store and retrieve product data.
  - **Python**: Used for backend development, handling API requests, and data processing.
- **API Communication**: RESTful API was used to ensure smooth communication between the frontend (React) and the backend (Flask).

---

3. Features and Functionalities

- **Interactive Chat Interface**:
  - The user interface allows users to type queries and receive responses in real-time.
  - The chatbot responds with relevant product names and prices based on the user query.
- **Product Search**:
  - Users can search for products in various categories like **electronics**, **books**, and **clothing**.
  - The search is processed by the backend, which returns product names and prices from the mock database.
- **Responsive Design**:
  - The chatbot interface is responsive and designed to work seamlessly across desktop, tablet, and mobile devices.
- **Session Management**:
  - Each user session is tracked, maintaining chat history for the duration of the session.

    - Users can reset the chat at any time to start a fresh session.
- **Backend Integration**:
    - The backend is designed to handle search queries and return relevant product data using Flask and SQLite.

---

4. Sample Queries and Results

Here are some sample queries and the corresponding results from the chatbot:

Sample Query 1: "books"

- **Bot Response**:
    - "Book: Python Programming - $15"
    - "Book: Data Science - $30"

Sample Query 2: "electronics"

- **Bot Response**:
    - "Laptop - $1000"
    - "Smartphone - $500"

Sample Query 3: "clothing"

- **Bot Response**:
    - "T-Shirt - $20"
    - "Jeans - $40"

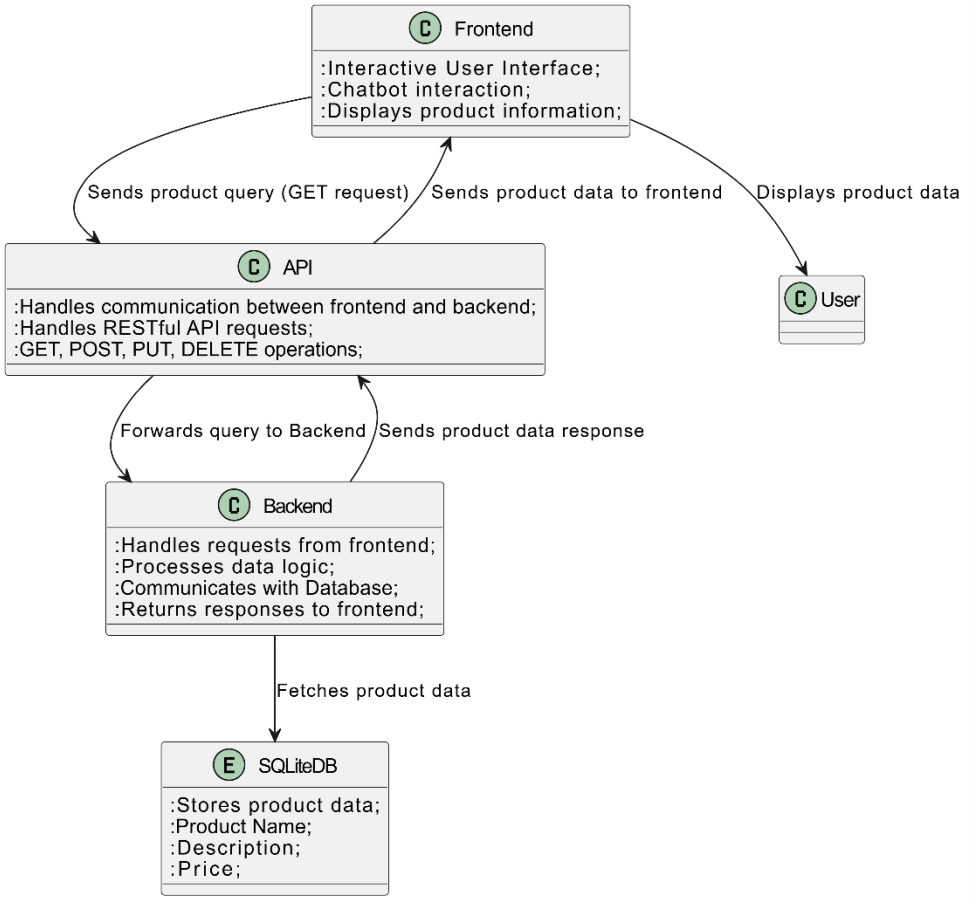
---

5. Architecture Diagram

The architecture diagram below shows the flow of data and interactions between the different components of the project.

Architecture Components:

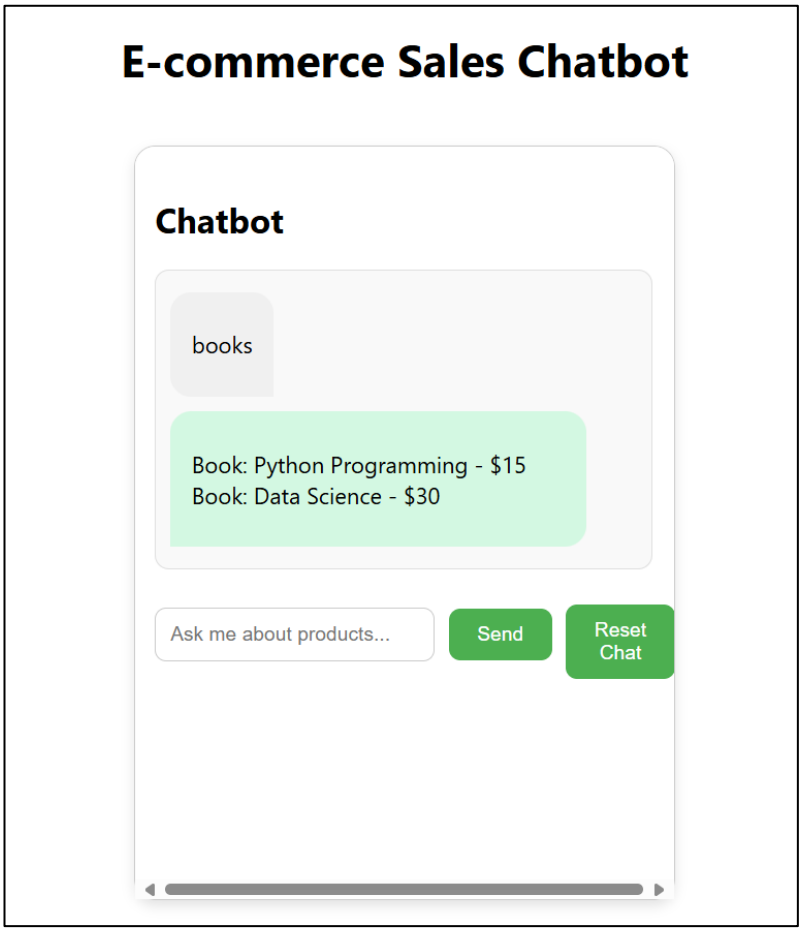
- **Frontend**: React.js is used to build the interactive user interface where users interact with the chatbot.
- **Backend**: Flask is used to process requests and retrieve data from a mock database.
- **Database**: A mock inventory system using SQLite stores the product data.
- **API**: RESTful API is used for communication between the frontend and backend.


Frontend
:Interactive User Interface;
:Chatbot interaction;
:Displays product information;
Sends product query (GET request)
Sends product data to frontend
Displays product data
API
:Handles communication between frontend and backend;
:Handles RESTful API requests;
:GET, POST, PUT, DELETE operations;
User
Forwards query to Backend
Sends product data response
Backend
:Handles requests from frontend;
:Processes data logic;
:Communicates with Database;
:Returns responses to frontend;
Fetches product data
SQLiteDB
:Stores product data;
:Product Name;
:Description;
:Price;

6. Screenshots

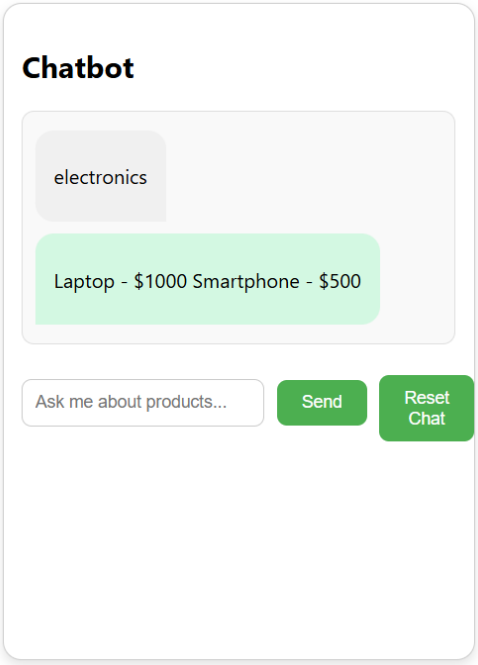
Here are some screenshots showing the **E-commerce Sales Chatbot** in action.

Screenshot 1: Chatbot Interface (User Interaction)



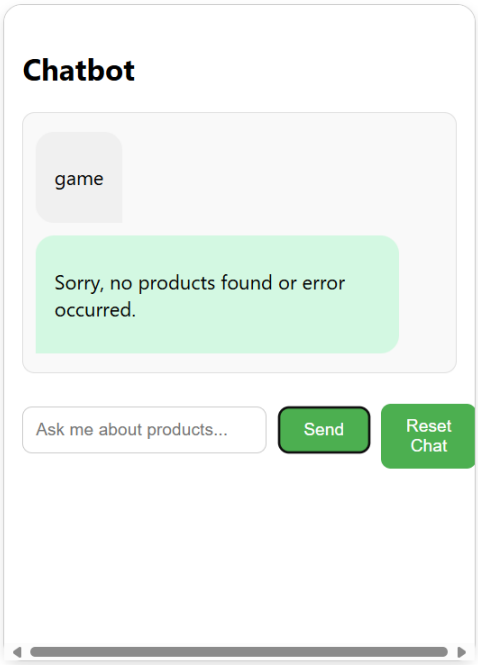
Description: The chatbot interface where users can type queries to search for products. It is responsive and works across different devices.

Screenshot 2: Search Results Display



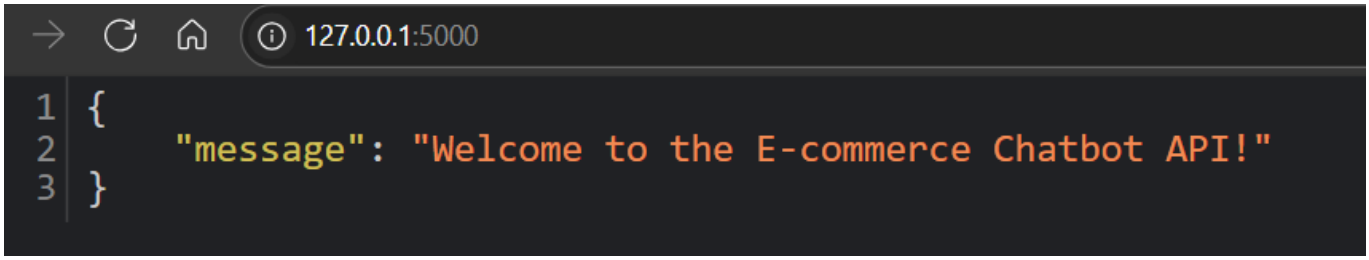
Description: The chatbot displaying relevant products based on the user's query (e.g., "electronics").

Screenshot 3: No Products Found (Error Message)



Description: This screenshot shows the chatbot's response when a user queries a non-existent product, like "game". The error message displayed is: "Sorry, no products found or error occurred."

Screenshot 4: API Response Message



This screenshot shows the API response when accessing the backend server at 127.0.0.1:5000. The message returned is: "Welcome to the E-commerce Chatbot API!"

---

7. Challenges Faced and Solutions

- **Challenge 1**: Handling empty search results (when no products match the query).
    - **Solution**: We implemented a fallback response stating "No products found" if the backend query returns no results.
- **Challenge 2**: Ensuring smooth communication between the frontend and backend.
    - **Solution**: We used Axios in the frontend to send GET requests to the Flask backend and handled responses with clear error management to ensure seamless data flow.
- **Challenge 3**: Maintaining session history and chat continuity.
    - **Solution**: React's state management was used to maintain chat history during the interaction, and local storage was used to persist sessions if needed.

---

8. Conclusion

The E-commerce Sales Chatbot project successfully delivers an interactive, user-friendly way for customers to search and explore products on an e-commerce platform. With the help of modern technologies like React.js for the frontend and Flask for the backend, the project provides a solid foundation for future improvements and scaling.