REPORT ON AI SALESBOT

1. Introduction

The AI-Powered Sales Assistant Chatbot is a web-based application designed to streamline product management and sales interactions for an e-commerce platform. It provides an intuitive chat interface for users to perform CRUD (Create, Read, Update, Delete) operations on products, enhancing efficiency and user experience in managing inventory and sales data

2. Technology Stack

Frontend:

- **React.js:** A JavaScript library for building user interfaces, chosen for its component-based architecture, efficient DOM updates (via Virtual DOM), and vibrant ecosystem.
- **Material-UI (MUI):** A popular React UI framework that implements Google's Material Design, providing pre-built, accessible, and customizable components for a modern and responsive user interface.
- **CSS-in-JS (Styled Components / MUI styled utility):** For styling components, offering dynamic styling capabilities and avoiding class name collisions.

- **JavaScript (ES6+):** The primary programming language for client-side logic.
- **HTML5 & CSS3:** Standard web technologies for structure and basic styling.

Backend:

- **Python:** The programming language for server-side logic.
- **Flask:** A lightweight and flexible Python web framework, suitable for building RESTful APIs. Chosen for its simplicity, extensibility, and minimal setup, making it ideal for rapid development.
- **Flask-SQLAlchemy:** An extension for Flask that integrates SQLAlchemy (a powerful SQL toolkit and Object-Relational Mapper) for database interactions.
- **Flask-Bcrypt:** For securely hashing and salting passwords, ensuring user data protection.
- **Flask-CORS:** To handle Cross-Origin Resource Sharing, allowing the frontend (running on a different origin) to communicate with the backend.
- **SQLite:** A lightweight, file-based relational database used for development and local storage of user, product, and message data. It's suitable for small to medium-sized applications and quick prototyping.

Database:

- **SQLite:** Chosen for ease of setup and local development. For production, this would typically be replaced with a more robust RDBMS like PostgreSQL or MySQL.

Development Tools:

- **npm/Yarn:** Package managers for JavaScript dependencies.
- **pip:** Package manager for Python dependencies.
- **Visual Studio Code (or similar IDE):** For code development and debugging.
- **Browser Developer Tools:** For frontend debugging and performance analysis.
- **Git/GitHub:** For version control and collaborative development.

3.System Architecture

The application follows a client-server architecture:

- **Frontend (Client):** A React.js single-page application (SPA) that provides the user interface. It communicates with the backend API via HTTP requests (e.g., fetch API) to send user commands and receive responses.
- **Backend (Server):** A Flask API that handles business logic, database interactions, and

authentication. It receives requests from the frontend, processes them (e.g., parses chat commands, performs CRUD operations), interacts with the SQLite database, and sends JSON responses back to the frontend.

- **Database:** SQLite database stores user credentials, product information, and chat history.

4. Key Features

- **User Authentication:** Secure login and registration functionality for users.
- **AI Sales Assistant Chat Interface:** An intuitive and interactive chat window where users can type commands.
- **Product CRUD Operations:**
- **Add Product:** Users can add new products with details like name, price, quantity, and category.
- **Update Product:** Users can modify existing product details (e.g., price, quantity, category).
- **Delete Product:** Users can remove products from the inventory.
- **Search Product:** Users can search for products by keywords.
- **List All Products:** Display all products currently in the inventory.

- **Filter by Category:** Users can view products belonging to a specific category.
- **Contextual Welcome Message:** A personalized welcome message is displayed upon successful login, guiding the user on available commands and functionalities.
- **Persistent Chat History:** Chat messages are saved to the database and loaded upon subsequent logins, providing continuity.
- **Responsive UI:** The frontend is designed to be user-friendly and adapt to various screen sizes using Material-UI.
- **Error Handling:** Robust error handling on both frontend and backend for user feedback and stability.

5. Database Schema (Simplified)

- **User Table:**
- id (Primary Key)
- username (Unique)
- email (Unique)
- password_hash
- **Product Table:**
- id (Primary Key)
- name

- price
- quantity
- category
- user_id (Foreign Key to User.id)
- **Message Table:**
- id (Primary Key)
- content
- role (e.g., 'user', 'assistant')
- timestamp
- user_id (Foreign Key to User.id)

6. Development Process and Approach

The project followed an iterative development approach, focusing on building core functionalities incrementally. Each iteration involved:

1. **Planning:** Defining the next set of features.
2. **Implementation:** Coding both frontend and backend components.
3. **Testing:** Manual testing of features, including UI interactions and API responses.
4. **Debugging & Refinement:** Addressing bugs, improving usability, and refactoring code.

Key focus areas included:

- Clear separation of concerns between frontend and backend.
- Modular component design in React for reusability.
- RESTful API design for clear communication.
- Prioritizing secure authentication practices.

7. Challenges and Solutions

- **Challenge 1: Welcome Message Display Loop:**
- *Problem:* The welcome message was appearing infinitely due to component re-renders triggering the message display logic repeatedly.
- *Solution:* Implemented a hasShownWelcome state flag in App.js. The WelcomeMessage component is now conditionally rendered only if this flag is false. Once the message is displayed, the flag is set to true, preventing subsequent renders in the same session. The flag is reset on logout.
- **Challenge 2: Input Field and Message Sending:**
- *Problem:* The chatbot was not taking user input, and messages weren't being sent due to mismatches in state variables and improper event handling.
- *Solution:* Ensured the StyledTextField's value and onChange props were correctly bound to the inputMessage state variable. Added null checks for event objects

in handleSendMessage to
prevent preventDefault errors on button clicks.

- **Challenge 3: Message Formatting (Line Breaks):**
- *Problem:* Messages, especially the welcome message, were appearing as single paragraphs despite attempts to include newlines.
- *Solution:* Applied whiteSpace: 'pre-wrap' CSS property to the MessageBubble styled component in App.js (and also to
the Message component's Typography element for consistency). This CSS property forces the browser to respect whitespace and newline characters within the text content.
- **Challenge 4: UI/UX Consistency:**
- *Problem:* Maintaining the original UI/UX design while integrating new features and fixing issues.
- *Solution:* Used Material-UI's styled utility and ThemeProvider to define consistent themes and component styles. Careful restoration of original component structures and prop usage.

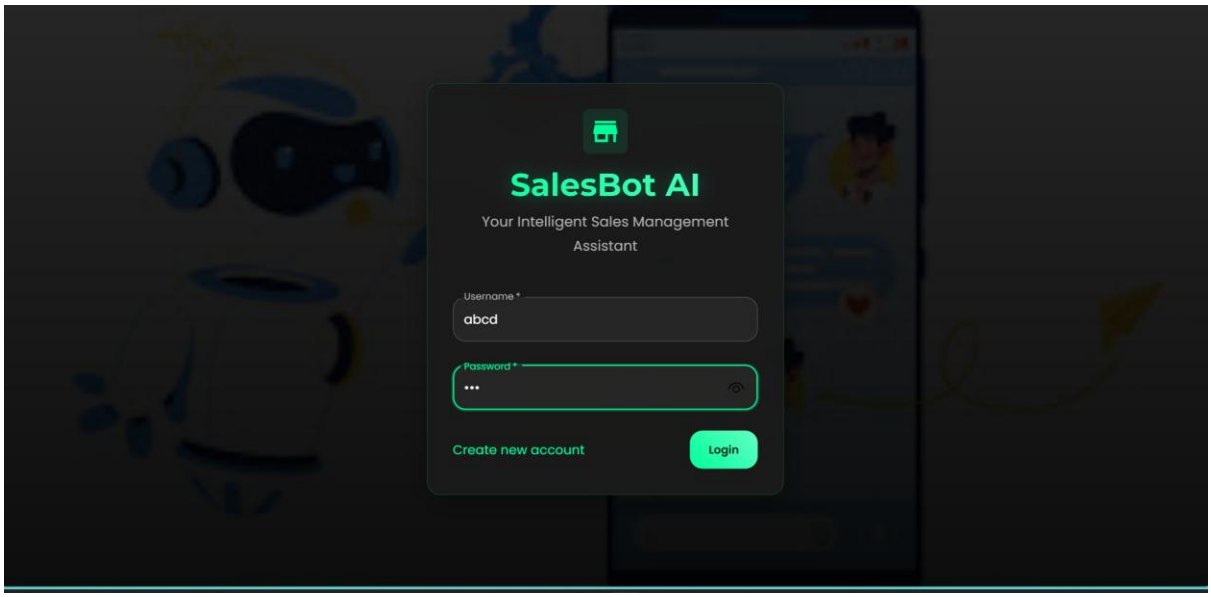
8. Future Enhancements

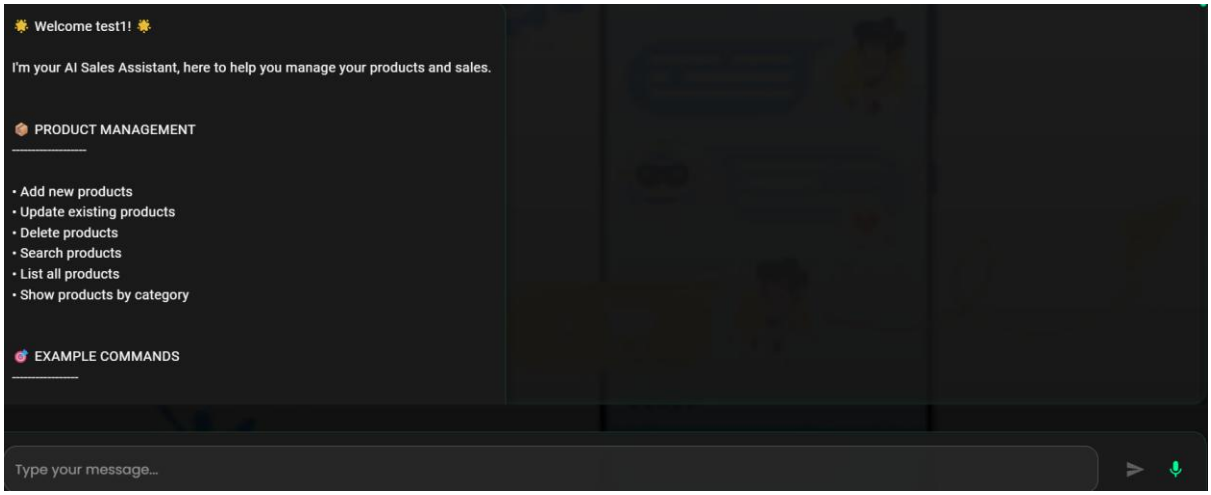
- **Natural Language Processing (NLP):** Integrate a more advanced NLP library (e.g., NLTK, SpaCy, or

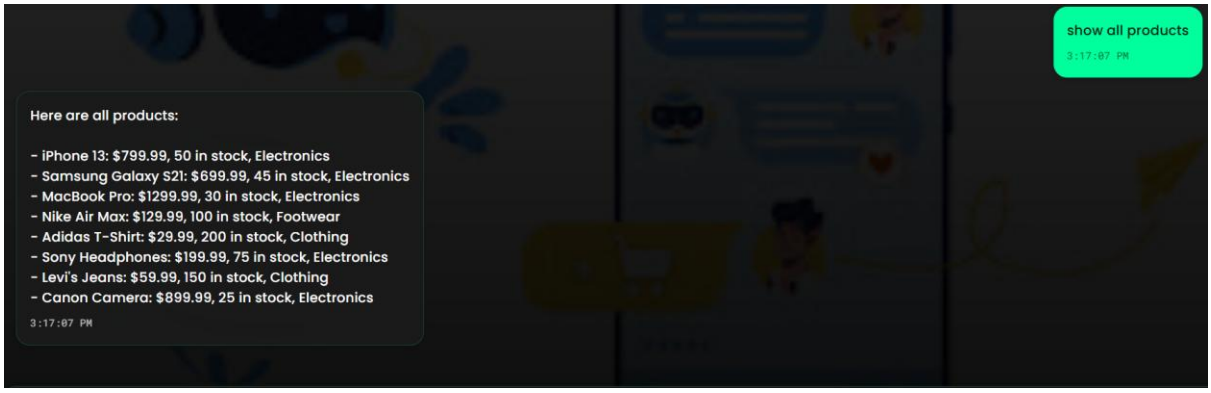
a cloud-based NLP API) to allow for more conversational and less command-driven interactions.

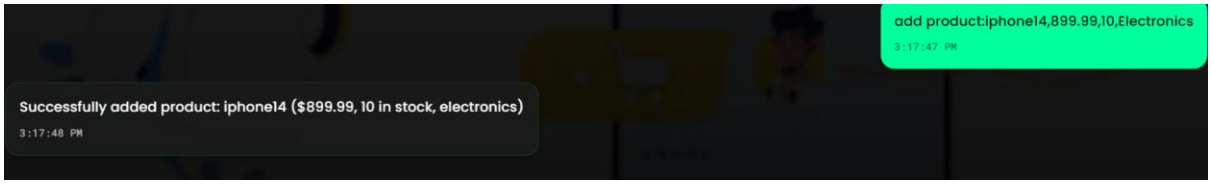
- **Advanced Analytics:** Implement features to track sales trends, popular products, and inventory levels.
- **User Roles & Permissions:** Introduce different user roles (e.g., admin, sales rep) with varying access levels.
- **Real-time Updates:** Use WebSockets for real-time chat and product updates (e.g., when another user modifies a product).
- **Cloud Deployment:** Deploy the application to cloud platforms (e.g., AWS, GCP, Heroku) for scalability and accessibility.
- **Unit & Integration Testing:** Implement automated tests for both frontend components and backend API endpoints.
- **Advanced Search:** Implement fuzzy search, filtering by multiple criteria, and sorting options.
- v Tools

OUTPUT :-


SalesBot AI
Your Intelligent Sales Management Assistant
Username *
abcd
Password *
Create new account
Login


Welcome test1!
I'm your AI Sales Assistant, here to help you manage your products and sales.
PRODUCT MANAGEMENT
• Add new products
• Update existing products
• Delete products
• Search products
• List all products
• Show products by category
EXAMPLE COMMANDS
Type your message...


show all products
3:17:07 PM
Here are all products:
- iPhone 13: $799.99, 50 in stock, Electronics
- Samsung Galaxy S21: $699.99, 45 in stock, Electronics
- MacBook Pro: $1299.99, 30 in stock, Electronics
- Nike Air Max: $129.99, 100 in stock, Footwear
- Adidas T-Shirt: $29.99, 200 in stock, Clothing
- Sony Headphones: $199.99, 75 in stock, Electronics
- Levi's Jeans: $59.99, 150 in stock, Clothing
- Canon Camera: $899.99, 25 in stock, Electronics
3:17:07 PM


add product:iphone14,899.99,10,Electronics
3:17:47 PM
Successfully added product: iphone14 ($899.99, 10 in stock, electronics)
3:17:48 PM


update product:iphone14,stock,14
3:18:30 PM
Successfully updated stock for iphone14 to 14
3:18:31 PM


Delete product:Iphone14
3:19:27 PM
Successfully deleted product: iphone14
3:19:27 PM