Designing and implementing a chatbox can be a complex task, but I can provide you with a high-level algorithm to get you started. Keep in mind that the implementation details may vary depending on the platform and technologies you're using. Here's a basic algorithm:

1. **Define the Requirements:**
   - Determine the purpose of your chatbox (e.g., customer support, virtual assistant, chatbot, etc.).
   - Specify the features you want, such as text input, messages, user authentication, etc.

2. **Choose a Platform:**
   - Decide whether you're building a web-based chatbox, mobile app, or integrating it into an existing platform.

3. **Select a Technology Stack:**
   - Choose the programming languages, frameworks, and libraries you'll use (e.g., Python, JavaScript, Node.js, React, etc.).

4. **User Interface (UI) Design:**
   - Create a user-friendly and visually appealing chatbox interface with message bubbles, input field, and any other elements you need.

5. **Backend Development:**
   - Develop the backend server to handle user interactions, process messages, and store chat history if necessary.
   - Implement natural language processing (NLP) if you want the chatbox to understand and respond to user messages effectively.

6. **Frontend Development:**
   - Build the frontend components for the chatbox, including the input field and message display area.
   - Implement real-time updates to display messages as they come in.

7. **User Authentication (Optional):**
   - If required, add user authentication and authorization to secure the chatbox.

8. **Database Integration (Optional):**
   - Set up a database to store chat histories and user data.

9. **Testing:**
   - Thoroughly test the chatbox for usability and functionality.
   - Test it with real users to gather feedback for improvements.

10. **Deployment:**
    - Deploy your chatbox on a server or hosting platform of your choice.

11. **Maintenance and Updates:**
    - Regularly update and maintain your chatbox to fix bugs, add features, and improve its performance.

12. **Scalability (Optional):**
    - Plan for scalability, especially if you expect a large user base. Consider load balancing and cloud-based solutions.

13. **Monitoring and Analytics (Optional):**
    - Implement monitoring tools and analytics to track user interactions and improve the chatbox's responses over time.

14. **Security:**
    - Ensure the chatbox is secure and protect user data. Implement encryption and follow security best practices.

15. **Documentation:**
    - Document your code and chatbox usage to make it easier for others to understand and work with.

This algorithm provides a general guideline for designing and implementing a chatbox. The specific steps and technologies you use may vary based on your project's requirements and constraints.