Design Document: Svelte Chatbot Application

Version: 1.0

Date: 2023-05-19

1. Introduction
    1. Purpose

       The purpose of this document is to provide a detailed design overview of the Svelte chatbot application. The application allows users to interact with an AI chatbot using various models and provides a user-friendly interface for seamless communication.
    2. Scope

       The scope of this document covers the architectural design, user interface, API communication, and proposed improvements for the Svelte chatbot application.
    3. Definitions, Acronyms, and Abbreviations
        - API: Application Programming Interface
        - UI: User Interface
        - NASA: National Aeronautics and Space Administration
        - IEEE: Institute of Electrical and Electronics Engineers
2. System Architecture
    1. Overview

       The Svelte chatbot application follows a client-server architecture. The client-side is built using the Svelte framework, which communicates with a server-side API to fetch available models and exchange chat messages.
    2. Client-Side Components
        - App.svelte: The main component that encapsulates the entire application.
        - Controls: Handles user input, model selection, and generating AI responses.
        - Chat History: Displays the conversation history between the user and the AI, with the most recent messages at the top.
        - Markdown Support: Renders the AI's responses using the svelte-markdown component.
    3. Server-Side Components
        - API Endpoints:
            - /api/chat: Handles the chat message exchange between the client and the AI.
            - /api/tags: Provides the list of available models for selection.
3. User Interface Design
    1. Layout

       The user interface consists of the following main components:
        - Title: Displays the application title "Chatter".
        - Controls: Contains the user input textarea and model selection options.
        - Chat History: Shows the conversation history between the user and the AI, with the most recent messages at the top.
        - Clear Chat: Users can clear the current chat and start a new conversation by clicking the "Clear Chat" button.
    2. User Interactions
        - Model Selection: Users can click on the available models to select the desired model for the conversation.
        - User Input: Users can enter their messages in the provided textarea.

- Generate Response: Users can generate an AI response by pressing Ctrl + Enter or clicking the generate button.
- Copy Chat History: Users can copy the entire chat history by holding the Ctrl key and clicking on the chat history area.
- Clear Chat: Users can clear the current chat and start a new conversation by clicking the "Clear Chat" button.
  - When not waiting for a response, the button displays "Clear Chat" and clears the chat when clicked.
  - While waiting for a response, the button displays "Chat In Progress" with the text bouncing left-to-right and right-to-left.

4. API Communication

   1. API Endpoints

      - /api/chat: Accepts POST requests with the selected model and user message, and returns the AI's response.
      - /api/tags: Returns the list of available models for selection.

   2. Request/Response Format

      - The API uses JSON format for request and response data.
      - The request payload includes the selected model and user message.
      - The response payload contains the AI's generated message.

5. Proposed Improvements

   1. Error Handling

      - Implement robust error handling for API requests and display user-friendly error messages.
      - Validate user input and provide feedback for missing or invalid data.

   2. Loading State

      - Disable user input and generate button during loading to prevent multiple requests.
      - Change the "Clear Chat" button to "Chat In Progress" with bouncing text during loading.

   3. Input Validation

      - Validate and sanitize user input to prevent empty or excessively long messages.

   4. Responsive Design

      - Optimize the application's layout and styling for various screen sizes using CSS media queries.

   5. Accessibility

      - Enhance the application's accessibility by adding ARIA attributes, keyboard navigation, and alternative text for images.

6. Conclusion

   The Svelte chatbot application provides a user-friendly interface for interacting with an AI chatbot. By implementing the proposed improvements, the application can be enhanced in terms of error handling, user experience, responsiveness, and accessibility. The design document serves as a guide for future development and maintenance of the application.