Assignment: Intelligent Chat Widget

Background

Modern websites require sophisticated customer engagement tools that can provide immediate, intelligent responses to user queries. By leveraging Large Language Models (LLMs) and a well-structured knowledge base, we can create an embeddable chat widget that delivers personalized, context-aware responses while maintaining conversation history for better user experience.

Objective

Design and implement a JavaScript-based chat widget that can be easily integrated into any website. The system should incorporate a knowledge base-driven LLM backend for intelligent responses, featuring a clean user interface for chat interactions and conversation history management.

Core Requirements

1. Widget Implementation
Create an embeddable JavaScript widget with the following properties:

- Goals: To provide a seamless chat experience that can be integrated into any website with minimal setup
- Actions: Initialize chat interface, manage connections, handle message events
- Integration: Simple script tag implementation with configurable parameters
- Styling: Customizable CSS to match host website's theme
- Responsiveness: Adaptive design for all screen sizes
- Eg script:

```
<script
  src="https://cdn.example.com/chat-widget.js"
  data-api-key="YOUR_API_KEY"
  data-theme="light"
  data-position="bottom-right"
  data-initial-message="How can I help you today?"
>
</script>
```

2. User Interface
Develop a chat interface with these features:

- Goals: To offer an intuitive and engaging chat experience
- Components:

    - Chat window with message history
    - Input field with send button
    - Conversation history viewer
    - Minimize/maximize controls
- Real-time message updates
- Loading states and typing indicators
- Error handling and retry mechanisms

3. Backend System
Implement a backend service with:
- Goals: To process user queries and provide relevant responses from the knowledge base
- Components:
    - Knowledge base management system
    - LLM integration for natural language understanding
    - API endpoints for widget communication
- Features:
    - Query processing and response generation
    - Conversation context management
    - Knowledge base updates and maintenance

4. Knowledge Base Integration
Implement a vector store-based knowledge base with:
- Goals: To maintain and serve accurate, up-to-date information
- Components:
    - Vector embedding generation system
    - Vector similarity search implementation
    - Document chunking and processing pipeline
    - Metadata management system
- Features:
    - Efficient semantic search capabilities

Nice-to-Have Additions

- Hybrid search capabilities (semantic + keyword)
- Rich media message support (images, links, files)
- User authentication and personalization
- Offline support with message queuing
- Styling: Customizable CSS to match host website's theme

Submission Guidelines

- Provide detailed technical documentation including:
    - Architecture overview
    - Implementation details

    - API Documentation
    - Integration guide

- Submit a GitHub repository containing:
    - Source code with clear directory structure
- Include clear setup instructions:
    - Development environment setup
    - Dependencies installation
    - Build and deployment procedures
    - Configuration options

Reference
https://www.intercom.com/live-chat