PART 3: AI INTEGRATION (CURIOSITY + OPTIONAL CODE)

Which model/framework you'd choose and why:

I would choose Google's Gemini AI model for its strong conversational abilities and natural language generation, making the chatbot smart and engaging. For the development framework, I'd use the Vercel AI SDK (@ai-sdk/react) within a Next.js application. The AI SDK simplifies connecting to AI models and handles complex tasks like streaming responses, which is crucial for a smooth chat experience.

How you would integrate it into a Next.js app:

Integration involves two main parts:

A server-side API route (like /api/chat). This route acts as a secure intermediary, receiving user messages from the frontend and forwarding them to the Gemini API, then streaming back the AI's response. This keeps API keys safe.

On the client-side, within the React component (like your ChatBot.tsx), the useChat hook from @ai-sdk/react is used. This hook connects to the API route and manages the entire chat state, including messages, input, and loading status.

How you'd handle input/output, and streaming responses:

Input: User messages are captured via a text input field, managed by the input state and handleInputChange from the useChat hook. When the user submits their message, the handleSubmit function sends it to the API route.

Output: The messages array provided by useChat contains both user and AI responses. This array is mapped over to display chat bubbles on the screen, with automatic scrolling to show the latest message.

Streaming Responses: This is handled automatically by the Vercel AI SDK. As Gemini generates its response, it sends it in small pieces (tokens). The SDK receives these pieces and continuously updates the AI's message in the chat, making it appear as if the AI is typing in real-time. A loading indicator (like your LoadingDots) is shown while the AI is responding, improving user experience.