Coding Exercise: AI-Powered Educational Chat Application

Overview

Candidates are required to develop a **Next.js/ReactJS-based interactive chat application** that simulates **real-time AI-powered educational discussions**. The application should provide **real-time updates, advanced state management, and optimized performance** while ensuring a seamless user experience.

To complete this assignment, candidates must demonstrate expertise in **React.js/Next.js, TypeScript, Zustand/Redux, WebSockets (frontend-only), and UI frameworks such as TailwindCSS or Material UI**.

Application Components

1. Real-Time Chat Interface

Objective:

Develop an interactive **educational chat interface** that ensures smooth user interaction and real-time updates.

Key Features:

- **Chat UI & Message Handling:**
  - Develop a **responsive chat interface** that supports **rich text formatting** (bold, italic, lists, links).
  - Implement **scroll-to-bottom behavior** for seamless navigation.
  - Ensure **message animations** using Framer Motion.
- **Real-Time Experience Simulation:**
  - Display **"Typing..." indicators** for AI-generated responses.
  - Show **sent, delivered, and read receipts**.
  - Implement **user presence indicators** (active, offline, typing).
- **Dark Mode & Theming Support:**
  - Enable **light and dark mode switching** with persistent settings.
  - Implement **custom theming** using CSS variables and Zustand.
- **Message Input Enhancements:**
  - Support **emoji selection, markdown-style formatting, and message editing**.
  - Enable **keyboard shortcuts** (e.g., `Shift + Enter` for new lines, `/help` for quick actions).
  - Display a **character limit indicator** for long messages.

---

2. API Integration & State Management

Objective:

Manage **frontend state efficiently** while simulating real-time API interactions.

Key Features:

- **Mock API Integration (Frontend Only):**
    - Use **mock API services or JSON files** to simulate data retrieval.
    - Implement **optimistic UI updates** for seamless user experience.
- **State Management & Data Persistence:**
    - Utilize **Zustand or Redux Toolkit** to manage:
        - **Chat history** (persisted across sessions).
        - **User session data** (mock authentication).
        - **UI preferences** (theme, accessibility settings).
- **Error Handling & Loading States:**
    - Implement **skeleton UI placeholders** for loading messages.
    - Add **graceful API failure handling**, including retry logic and notifications.
    - Provide a **network reconnect indicator** for unstable connections.

---

3. Performance & UI Optimization

Objective:

Ensure **fast performance, accessibility, and SEO optimization** for improved user engagement.

Key Features:

- **Next.js Performance Enhancements:**
    - Use **Server-Side Rendering (SSR) or Incremental Static Regeneration (ISR)**.
    - Optimize **lazy loading, tree-shaking, and code-splitting**.
- **SEO & Accessibility Compliance:**
    - Implement **dynamic meta tags** for improved discoverability.
    - Ensure **ARIA compliance**, keyboard navigation, and high contrast modes.
    - Integrate **speech-to-text and text-to-speech features** for accessibility.
- **Micro-Interactions & Animations:**
    - Use **Framer Motion** for smooth transitions.

    - Implement **message fade-in effects** for an enhanced visual experience.
    - Add **hover effects and button animations** to improve user interaction.

Tools & Libraries (Choose Any)

Candidates should select and justify their technology stack choices.

Frontend Stack:

- **Framework:** Next.js/ReactJS
- **State Management:** Zustand / Redux Toolkit
- **Styling:** TailwindCSS, Chakra UI, or Material UI
- **Animations:** Framer Motion for UI transitions
- **WebSockets/Polling:** Native WebSocket API (mocked for frontend testing)

Bonus Features (Optional)

- **User Mentions (`@username`)** – Highlight messages mentioning specific users.
- **Voice Message Support** – Enable **recording and playback of voice messages**.
- **Chatbot Personality Settings** – Allow users to **customize AI behavior and response style**.
- **PWA Support** – Implement **offline mode and installable chat functionality**.

Notes:

- The application should be **designed for scalability and efficiency**.
- Proper **error handling and logging** should be implemented.
- The choice of **state management, UI libraries, and optimization strategies should be justified**.