Web Application Developer Intern Assignment

Objective: Build a real-time chat application using the MERN Stack (MongoDB, Express, React, and Node.js). This assignment will help assess your skills in backend development, API design, real-time data handling, and your understanding of full-stack application architecture.

Assignment Requirements:

1. **User Authentication**:
    - Implement user registration and login using JWT (JSON Web Tokens) for session management.
    - Store user details securely in MongoDB.
    - Ensure proper validation and hashing for sensitive data like passwords.
2. **Chat Functionality**:
    - Allow authenticated users to send messages to each other in real-time.
    - Use WebSockets (preferably with Socket.io) to enable live chat without refreshing the page.
    - Save chat history in MongoDB so that users can view past conversations when they log back in.
3. **User Interface**:
    - Create a basic frontend using React for sending and receiving messages.
    - Display a list of users online (who are currently connected).
    - Provide a simple chat UI with an input field for typing messages and a display area for chat messages.
4. **Online Presence Indicator**:
    - Show users as “online” or “offline” based on their connection status.
    - Use WebSockets to broadcast user status updates (when they log in or log out) to other connected clients.
5. **Basic UI Features**:
    - Ensure the chat window automatically scrolls down to show the latest messages.
    - Add timestamps for each message to display when it was sent.
    - Use a minimalist design, focusing on functionality.

Technical Requirements:

1. **Backend**:
    - Use **Node.js** and **Express** to set up the backend server.
    - Set up RESTful API endpoints for user authentication and messaging functionality.
    - Use **MongoDB** to store user profiles and chat history.
    - Use **Socket.io** for real-time communication.
2. **Frontend**:

    - Use **React** to build a simple UI for the chat application.
    - Use **Socket.io-client** on the frontend to connect to the WebSocket server.
3. **Database**:
    - Use MongoDB as the database to store user profiles and chat messages.
    - Design a basic schema for users (e.g., username, email, password) and for messages (e.g., sender, receiver, message content, timestamp).
4. **Other**:
    - Use Git for version control and submit a link to the repository (GitHub, GitLab, etc.).
    - Provide setup instructions in a README file, including how to install dependencies, set up environment variables, and run the application locally.

Bonus Features (Optional but Appreciated):

- Implement a "typing..." indicator that appears when the other user is typing.
- Add media message support (image upload) and store the images in the database or on a cloud service.
- Implement message read receipts, showing when a message has been seen by the recipient.

Evaluation Criteria:

1. **Code Quality**: Readable, modular, and well-documented code.
2. **Functionality**: The application should work as specified, with user authentication, real-time messaging, and user presence indicators.
3. **Technical Design**: Proper database schema design, REST API structure, and WebSocket integration.
4. **UI and UX**: While design complexity is not expected, the app should be user-friendly and functional.
5. **Documentation**: Clear instructions on how to set up and run the project, along with any assumptions or considerations made.

Submission:

- **Deadline**: within 3 to 4 Days
- **Submission Format**: Link to the Git repository with a README file that includes setup instructions and also hosting on Netlify.