Chat Application Development

Objective:

The goal of this task is to develop a real-time chat application where users can sign up, connect with other users, and engage in **one-on-one** or **group chats** with essential chat features, including **lazy loading, pagination, unseen message count, and notifications**.

Project Modules & Features

1. User Signup & Authentication

- Users should be able to **sign up** using an email and password.
- Implement **JWT-based authentication** for session management.
- Validate user credentials and ensure secure storage using **bcrypt hashing**.

2. Contact Module

- Users can see a **list of registered users** in a "Contacts" section.
- Clicking on a contact will initiate a **private chat**.
- **Unseen message count** should be displayed as a **badge** next to the contact name.

3. One-to-One Chat

- Users can send and receive messages in real-time.
- Messages should be **stored in the database**.
- Implement **WebSocket** for instant message updates.
- Add **read receipts** (seen/unseen ticks).
- Users should be able to **delete messages** (single message or entire chat).
- **Lazy loading & pagination**:
    - Older messages should load when scrolling up.
    - Pagination should be implemented to load messages in **batches** (e.g., 20 messages per request).

4. Group Chat

- Users can create/join group chats.
- Group members should be able to send and receive messages.

- Display the list of group members in the chat.
- WebSocket should handle group message updates.
- **Unseen message count per group chat** should be displayed.

5. Unseen Messages Highlighting

- If a user has **unseen messages**, show a **notification badge** next to the contact's name.
- When the chat is opened, mark the messages as **seen**.

6. Notifications

- Whenever a message is sent, the receiver should receive:
    - **Email notification**.
    - **Push notification**.
- Notifications should be sent only if the user is **not active** on the chat screen.

7. WebSocket Integration

- Implement WebSocket to ensure **real-time updates** without page refresh.
- Use WebSocket for:
    - Instant message delivery.
    - Real-time status updates (online/offline, typing...).
    - **Unseen message count updates**.

8. Lazy Loading & Pagination

- **Lazy Loading**: Older messages should load **only when the user scrolls up**.
- **Pagination**:
    - Load messages in **batches** of 20.
    - API should support **offset-based or cursor-based pagination**.

Technology Stack

Feature	**Technology**
Frontend:	React.js **(preferred)**

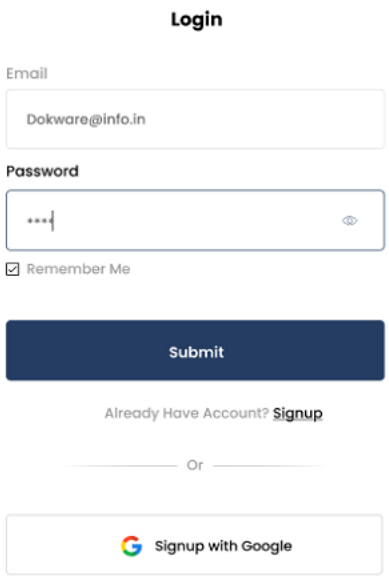
Backend:	PHP Laravel / Python (**preferred**)
Database:	My SQL (**preferred**)
Authentication:	JWT (JSON Web Token)
Real-Time Updates:	WebSocket
Push Notifications:	Firebase Cloud Messaging (FCM)

Note:- You are free to choose your technology.

Deliverables

- **Code Repository** (GitHub/GitLab/Bitbucket)
- **User Guide** (Basic UI Navigation)

Reference design



Timeline: 2 weeks

Note:- We understand that you may not have fully completed the assignment within the timeline, but we would still love to see your progress and approach. Our main focus is on evaluating how you think, problem-solve, and structure your work.