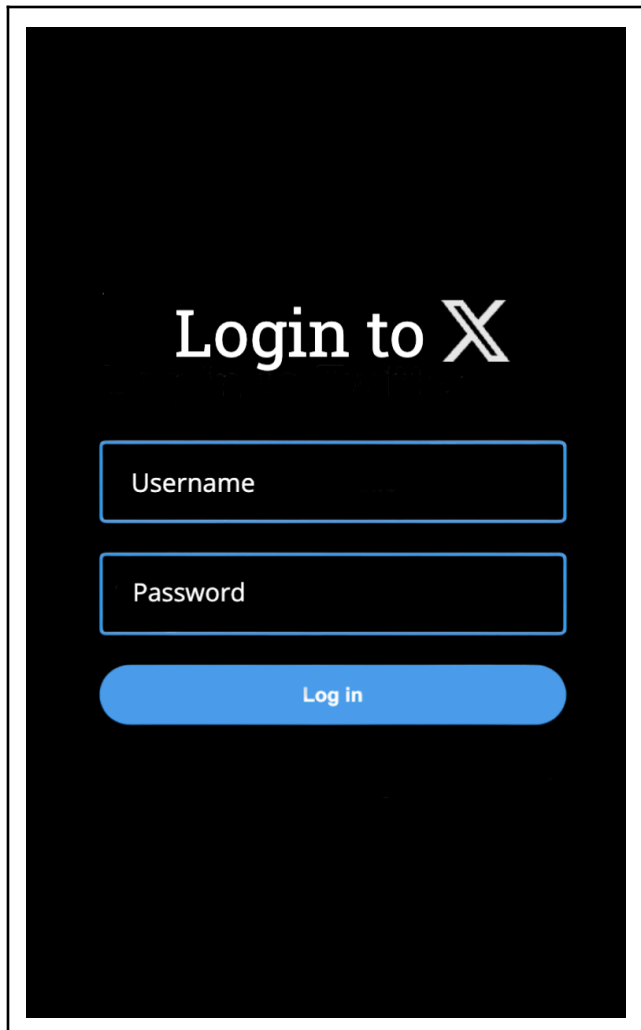


# Assignment - 2

## Problem Statement

Develop a backend with **Express & Socket.IO** and integrate it with the frontend. The backend should enable the following operations for a functioning **X** web application. The users will be able to do the following operations :

1. **Login:** Users should be able to enter the username and password to login to the application.
  - a. Add specific details here
    - i. Use **username** and **password** for login
    - ii. Hardcode a set of **N users** in your nodejs server. Only these credentials will be allowed to login in the application.
    - iii. No register functionality or DB connection needed.
  - b. **API endpoint (POST) :** **/api/user/login**
  - c. **UI :**

A login form for a web application. The form is centered on a black background. It features the text "Login to X" in a large, white, serif font. Below this text are two input fields: "Username" and "Password", both with white borders and light blue backgrounds. At the bottom of the form is a blue, rounded rectangular button with the text "Log in" in white.

- d.
- e. The User data must be stored in the JSON format with the following fields

Unset

```
{
  id : "",
  user_name:"",
  user_email_id:""
  password:"",
  profile_url:""
}
```

## 2. Landing Page

### a. List all the posts

- i. Posts have the following features - images, videos text, etc
- ii. This data should come from an API.
- iii. Api must be paginated
  1. Use the limit for this as **5** posts per request.
  2. On page load, 5 posts must be loaded on the DOM. (Assuming page-size=5 and initially page-number=1)
  3. Pass page-number and page-size as params accordingly.
  4. When we scroll and reach the bottom of the page, again paginated-api must be invoked to load 5 more posts in the DOM.
- iv. **API Endpoint (GET) : /api/posts**
- v. Infinite scroll features must be implemented.

### b. Create a new post

- i. Ability to add new post
  - ii. API integration
  - iii. **API Endpoint (POST) : /api/posts**
  - iv. The post functionality must also support the videos and images
  - v. On clicking post
    1. Should update the backend through API
    2. Automatically show up at the top of the list of posts
  - vi. Handle errors accordingly
    1. Button should be disabled if nothing is typed or added
    2. API validation must be there
    3. Show the errors through the alert wherever required
- (OPTIONAL)**

### c. Chat page

- i. When clicking over the message icon it should navigate to the messaging page
- ii. On initial load :
  1. List of all active users on the chat page, and the first chat must be selected by default
  2. If no one is active, show no active users screen (**only a side black screen displays no active users**) Style according to yourself.
  3. Users should be able to chat with anyone from the list.
  4. The **User Name** of users should be from what was used during login
- iii. Users should be able to talk to any other online person, messages should only go to them.

- iv. Whenever a new user comes-in then the online users must get updated in the UI and the users must be able to communicate with them.
- v. Reference Images has been added below

**d. Extra Requirement**

- i. **Create an endpoint to retrieve a post by its unique identifier.** *This endpoint will not be integrated into the frontend.*
- ii. **API Endpoint (GET) :** **/api/post/?id**

Method	URL	Description	Request body or Params
GET	/api/posts	Get paginated posts.	pageNumber and pageSize
POST	/api/posts	Create a new post.	{ id: 1, title: "Post 1", content: "Content 1" }
POST	/api/user/login	Login a user.	{ email: "test@example.com", password: "password123" }
GET	/api/post	Get post by id	id

## Hints / Guidelines:

### 1. Paginated API (Infinite Scroll):

- Use the limit for this as **5** posts per request.
- On page load, 5 posts must be loaded on the DOM. (Assuming page-size=5 and initially page-number=1)
- When we scroll and reach the bottom of the page, again paginated-api must be invoked to load 5 more posts in the DOM.

### 2. Chat/Socket Integration using socket.io:

- Utilize socket.io for real-time bidirectional communication between the server and clients.
- Implement socket.io's event-based messaging system to facilitate instant messaging between users.
- The messaging screen must be accessible from the navigation-bar.
- Whenever, a logged-in user opens the messaging screen. It must -
  - i. Broadcast a socket message stating it joined the chat. Thus, all other users at the messaging screen should listen for this and populate the user in user-list. (i.e. a new user is now available to chat)
  - ii. The newly joined user must also mark itself online by storing its details in the nodejs server (in-memory). This will help you to keep a track of all the online users and manipulate the user-list accordingly.
- Whenever someone clicks on a user from user-list, the chat window must be opened at the right. Enabling you to chat with the selected user.

- Messages in the chat should be visible to both the users in the chat and need not be persisted. (No need to store them)

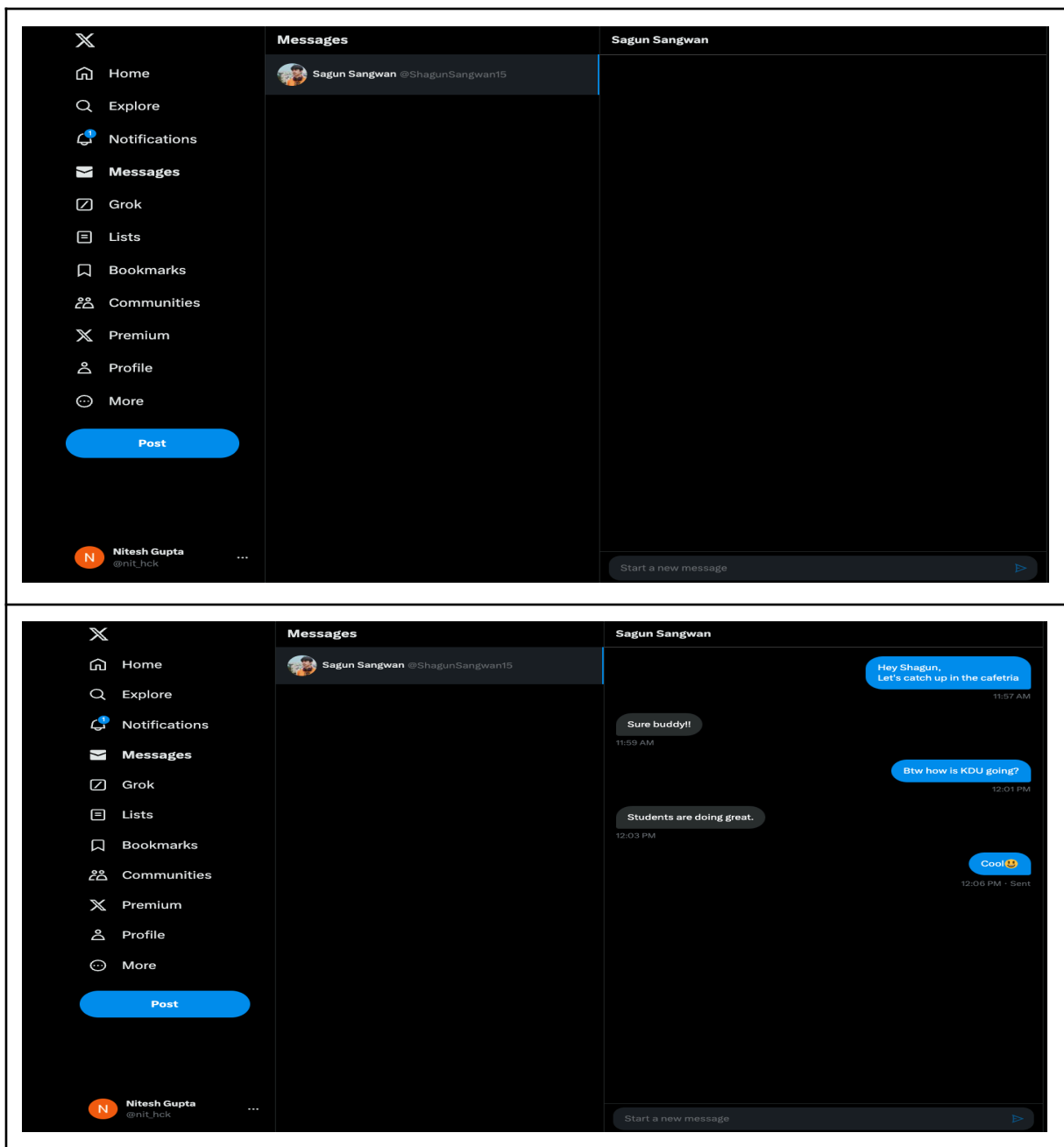
## Bonus Goal

In the chat for each message there must be a dynamic time change feature.

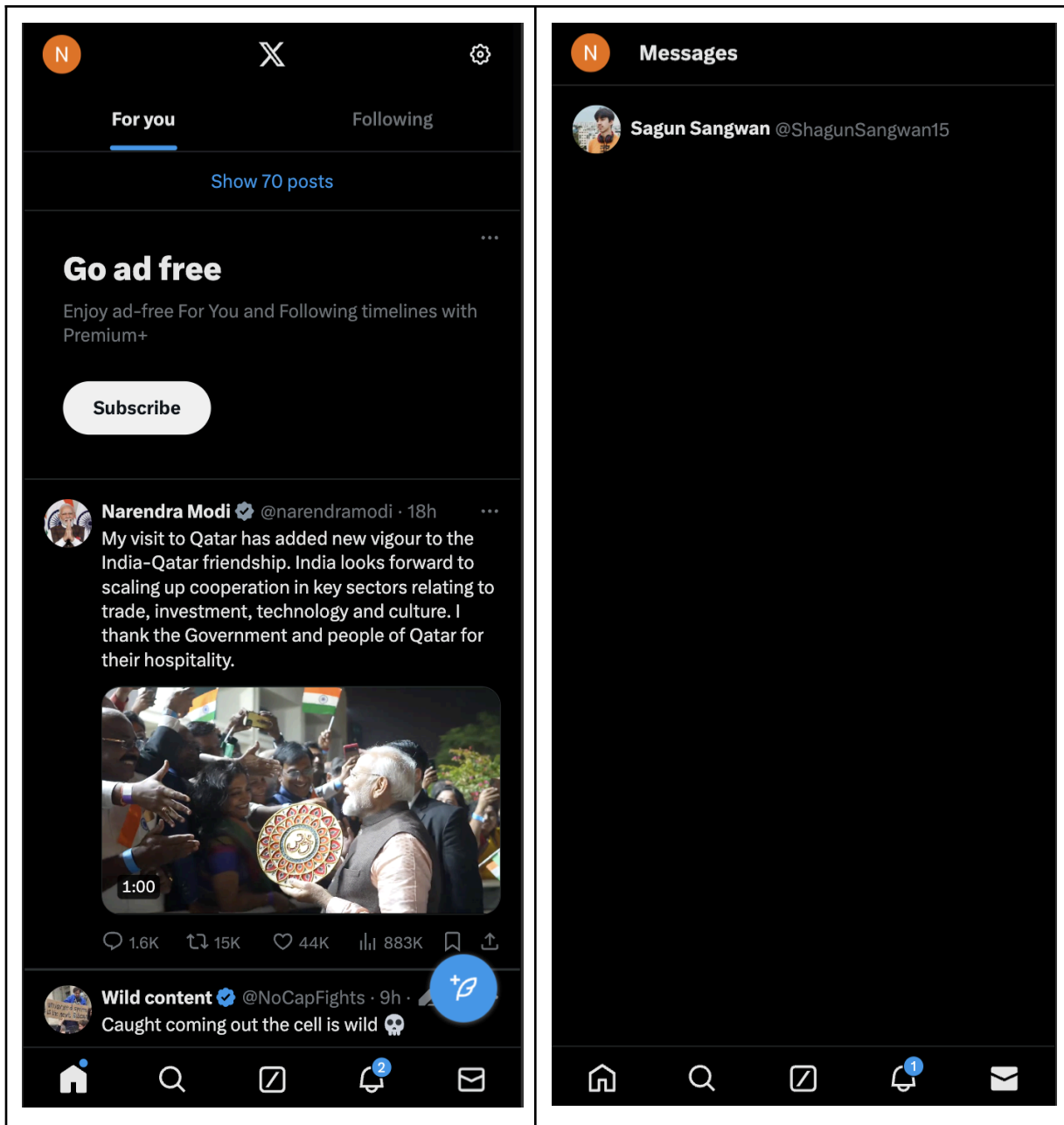
- Use the `Moment JS` package or similar and display the time for every chat-bubble.
- If the message is less than 10 minutes, show a relative time like “3 mins ago”.
- If it is greater than 10 mins, show the actual timestamp.

## Reference Snapshots

### Desktop View



## Mobile View





Sagun Sangwan

Hello nitesh!!

7:38 PM

Hey, ssup?

7:38 PM · Sent

Start a new message

