Task 3:

We are creating an MVP for a twitter clone, assuming the MVP will allow users to:

* Log in and out
* Write TweetClones
* View a chronologically ordered timeline of other user’s TweetClones
* Like Tweetclones

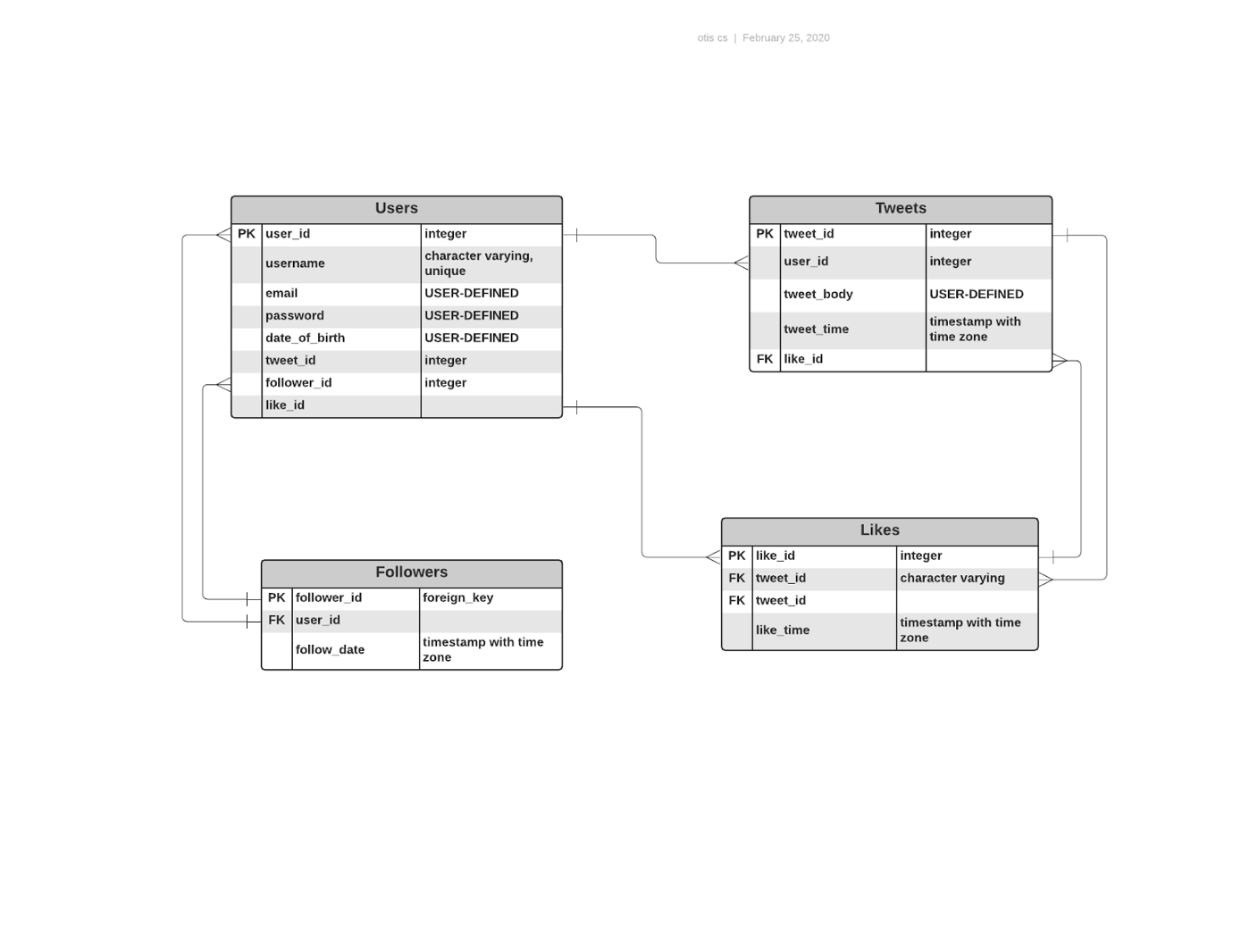
Please answer the following questions

1. How would you structure the data, what data structures are required?

Initially via SQL tables, with a structure similar to:   
(see diagram below for ease)

* User
  + user\_id
  + username
  + email
  + password
  + tweet\_id
  + follower\_id
  + like\_id
* Tweets
  + tweet\_id
  + tweet\_time
  + tweet\_body
  + user\_id
  + like\_id
* Followers
  + follower\_id (foreign\_key → user\_id)
  + user\_id
  + follow\_time
* Likes
  + like\_id
  + tweet\_id
  + user\_id
  + like\_time

Relationships  
User → 1 to many → Tweets  
User → many to many → Followers  
User → 1 to many → Likes  
Tweets → 1 to many → Likes



1. How would you feed the data to a frontend application?

Using the following or something similar.

users\_followed = []

for follower in followers:

   if follower.user\_id == user\_id:

        users\_followed.append(follower.follower\_id)

returned\_tweets = []

for tweet in tweets:

  if tweet.user\_id in users\_followed:

      returned\_tweets.append({"tweet\_id": tweet.tweet\_id,

                            "tweet\_time": tweet.tweet})

# Then sort the dict object:

for key, value in returned\_tweets.items():

  if tweet.user\_id in user.users\_followed:

    returned\_tweets.items()

    sorted\_returned\_tweets = returned\_tweets.sort(tweets.tweet\_time)

1. What endpoints would you define?

For part 3:

Aside from repeating my answer from part 1, I’m unsure.