Tinder KW

Basic Understanding

The Basic Functionalities of the Dating App that you need to include in the system would be

- Sign up & Login to the App
- User Able to only view, swipe left (pass) and swipe right (like) 10 other dating profiles in

total (pass + like) in 1 day.

- Same profiles can’t appear twice in the same day.
- User Able to purchase premium packages that unlocks one premium feature of your

choosing. A few examples:

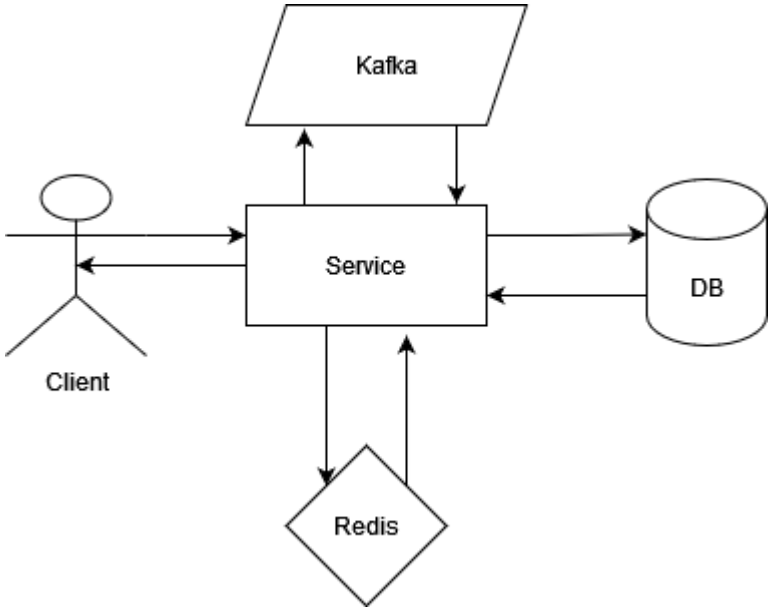
- No swipe quota for user
- Verified label for user

Implementation Scope

Phase	Objective
v1.0 (Functional)	1. Sign Up feature 2. Sign in feature
Next Iteration (Non-Functional)	1. Get Profile List feature 2. Verifying User feature 3. Activate Premium feature 4. Swapped Liked Counter feature

Design System

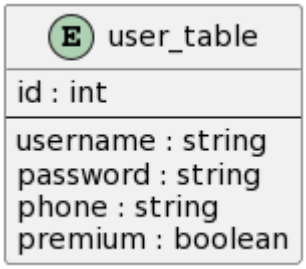
High level diagram is for non engineer purpose, the system will be easily introduced to non-engineer as we can easily show this diagram to them.



Client as a requester will have to request via Exposed API to the service, the requirement is we have data from the database, redis, and Kafka (will TBD if the data request is over 1m rpm) that will be providing data for service. when client needed user data after sign in the data can be acquired by token key to Redis so the data gathering will be fast, as for Kafka we can have the swipe feature to be in message queue because it will be fast and many people will do the swipe and like thing but it is not a rush things so we can later assign the total later in the service, and client need to hit the service via http server.

Functional Requirements

ERD - User Table



Sign Up - Post

For signing up users/front-end needed to hit the exposes API above using the contract needed, if the data is verified and username not currently in use, this service will making a new user and insert it to the user database, as that if its

success it will trying to generate JWT token and save it to the Redis for later use in the actual Apps. and for the response if all the process above OK (200) it will give JWT.Raw token, that will be used in the actual Apps.

```
// API call - by client from front-end
POST <authentication-service>/sign_up

Body:
{
  "username" : "abc",
  "password" : "123sssdddd",
  "phone" : "+622222222" //Optional
}

Respnonse:
200 OK
{
  "result" : {
    "token" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ..."
  }
}

400 Bad Request
{
  "result" : {
    "others" : "username already taken"
  }
}
```

Sign In - Post

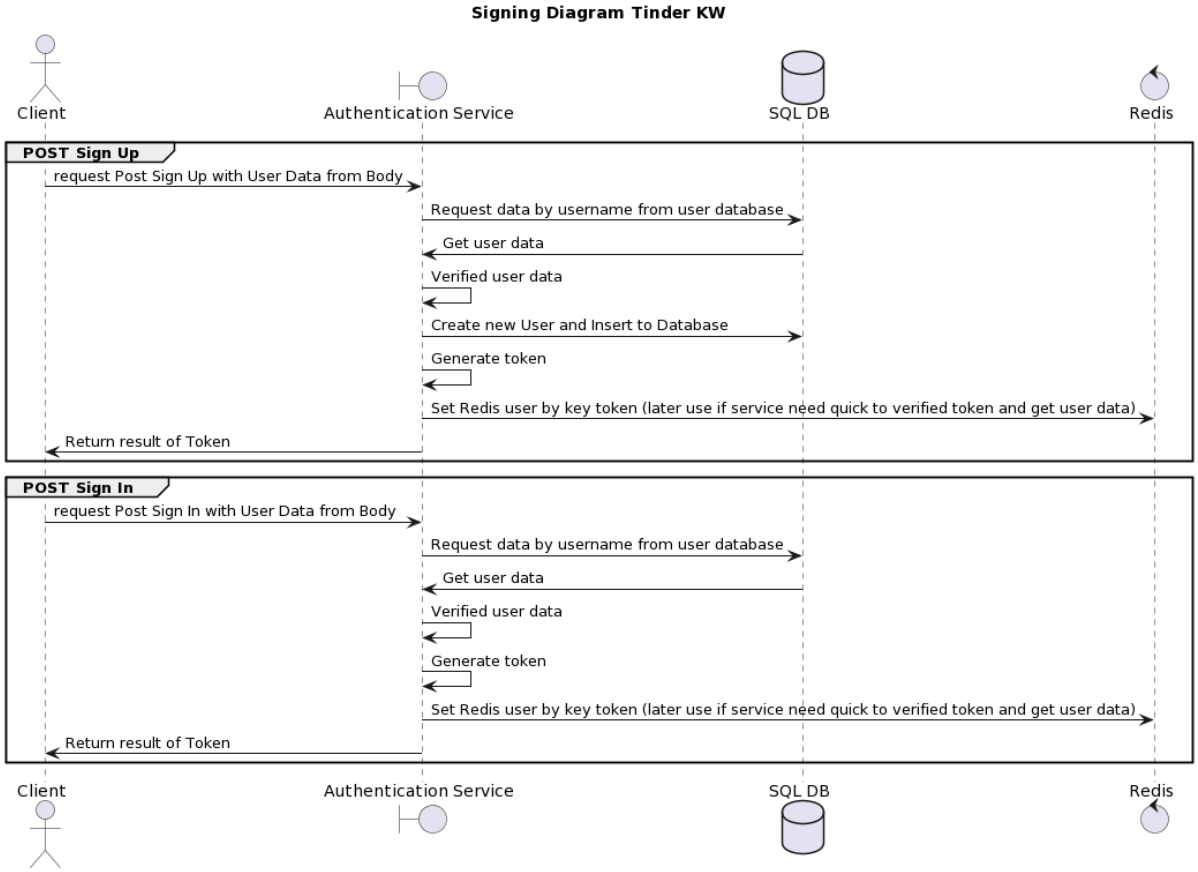
For signing in users/front-end needed to hit the exposes API above using the contract needed without the optional, if the data is verified and username there in the Database, this service will try to generate JWT token and save it to the Redis for later use in the actual Apps. and for the response if all the process above OK (200) it will give JWT.Raw token, that will be used in the actual Apps.

```
// API call - by client from front-end
POST <authentication-service>/sign_in

Body:
{
  "username" : "abc",
  "password" : "123sssdddd"
}

Respnonse:
200 OK
{
  "result" : {
```

```
    "token" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz..."
  }
}

500 Wrong Password
{
  "errors" : "user or password wrong"
}
```

Detailed Diagram

Detailed system will provide deep understanding for what this application do for sign up and sign in.



Non Functional Requirements

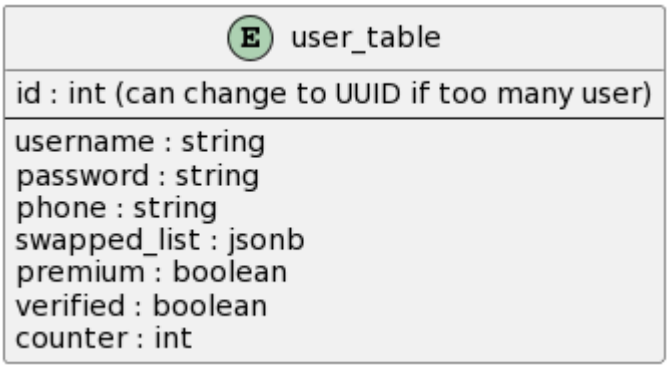
- Verified label for user
- Swap Quota max 10 / Unlimited for Premium user
- Same Profiles can’t appear twice

Assumption : There will be unlimited user already registered (for the purpose of simplification feature or maybe upgrade for TBD)

The verified user will needed another column in the database as today only have (id, user, pass, phone, premi_status) we will need more column verified (boolean) and to be verified we will need a service/API to be a feature to active it, same as premium status we also need that, and as for the swap it will handled by the front-end, but the client or front-end will need to actively sending if the user have swipe or like the profile, because this will be heavy we actually serve it right away we will making the swip/like feature by kafka, so the feature needed will be :

1. Get Profile List : for the first and every 10 profile list, client will request this feature to get another profile so its lighter for the backend if there is too many request at the same time, because we will implement almost like lazy load feature for this.
2. Swapped Liked Counter : this feature will be active when either the application closed, or the see profile list end / renew as the new request happen and send the list to swapped_list. the new profile will need this counter checked every time the new request happen because of this the RPM will be high for this so this will be handled by kafka.
3. Verifying User : if the user want to have verified badge the user need to click a verifying feature, by verifying the phone number, i choose phone number because its limited and literal premium to get new number over email that can be made in a second.
4. Activate Premium : premium will need to be verified user and as a premium the get profile list will negate the effect of Swapped Liked Counter, and more cool feature in the future 🙂

New ERD




```
swapped_list : []user_id (UUID/Int)
```