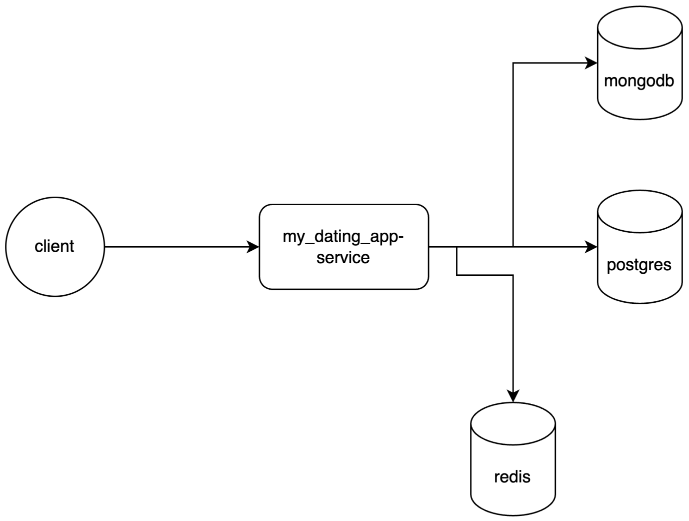
Imam Aprido Simarmata (+6282249384331/isimarmata09@gmail.com)
Software Engineer - Backend (**Dealls**)
25 May 2024

1. System Design & Tech Stack



Based on the requirement, here are the entities I should have on the database:

a. User,
b. Subscription (including available packages)
c. Activities (like & pass)

Below, I choose to have 3 database engines: MongoDB, PostgreSQL, and Redis.

MongoDB
I will use MongoDB to store activities. Since ‘activities’ is the main feature of a dating app, we must anticipate a high read/write and a fast-growing data size on its storage – NoSQL was made for that.

PostgreSQL
This SQL engine, with its attribute of ACID, is suitable for storing user and subscription data.

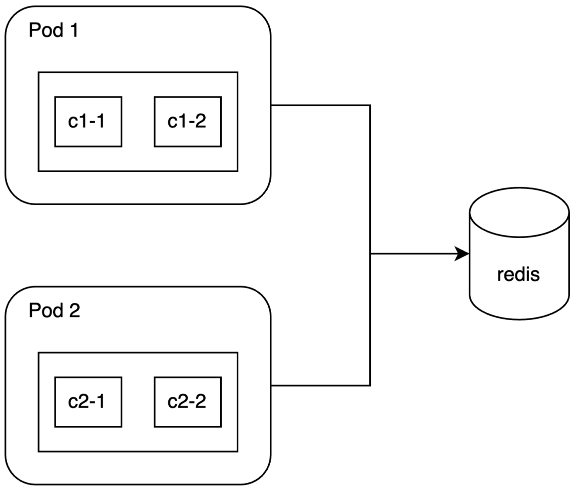
Redis

Caching using Redis avoids the frequent pulling of data to the main database.

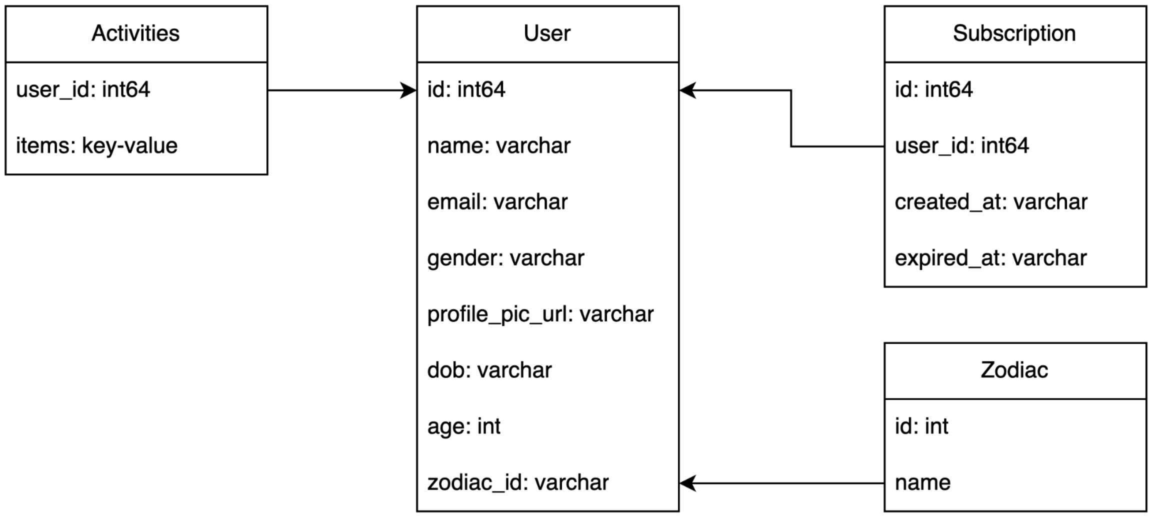
a. storing users' daily activity count,
b. storing profile details for feeds
c. plus, serving a locking mechanism (redsync) for subscriptions, preventing race conditions while updating user activities

Redsync

In case you deploy multiple machines/containers per service, a **distributed locking mechanism** is required.



2. ERD



Activities

We will only have one MongoDB document per user for storing their activities. Their activities will be stored as a key-value data structure (`activities.items`), with the date of activities going to be the key and the value will be the activities list of the respective date.

Subscription (unlimited pass)

SQL query to tell whether a user is an active subscriber or not:

```
SELECT EXIST(*) FROM subscription WHERE subscription.created_at
<= NOW() and subscription.expired_date >= NOW()
```

User

F/M (Female/Male) going to be the value of the user's gender.

3. Functional & ERD

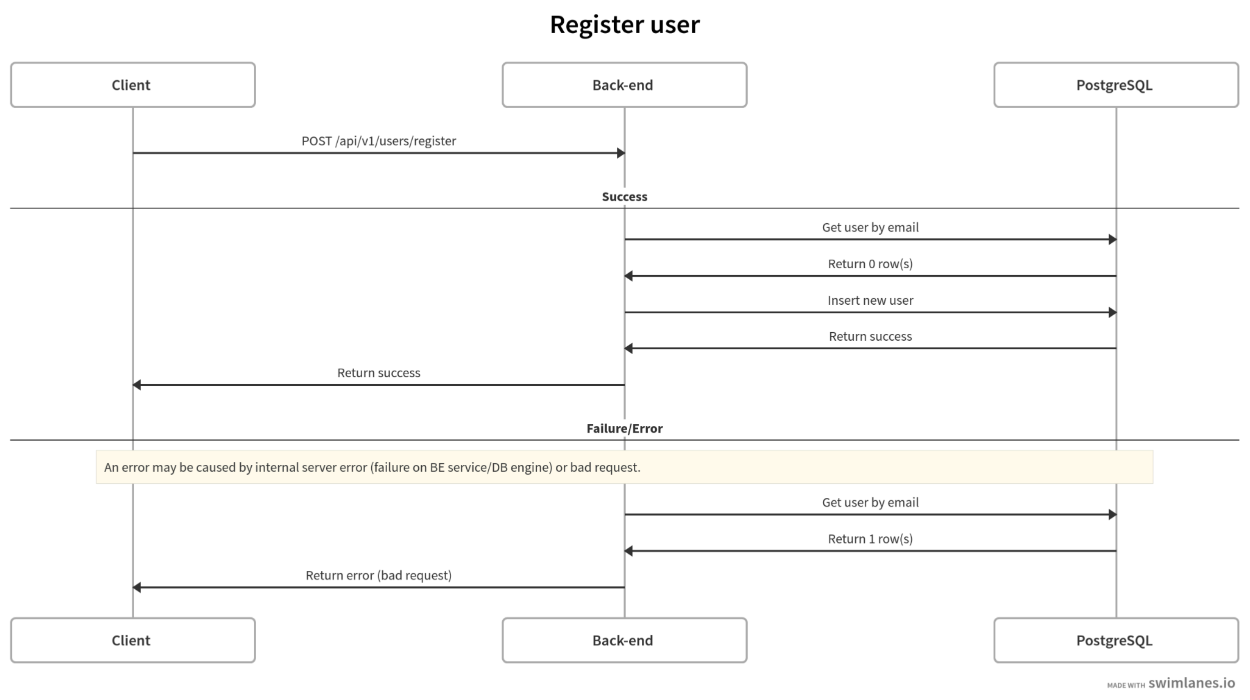
Register

Endpoint	`POST /api/v1/users/register`

Request body	name: string required
	dob: string required dd/mm/yyyy
	email: string required
	gender: string required
	
	{
	  "name" : "Imam Aprido Simarmata",
	  "dob" : "03/04/2001",
	  "email" : "isimarmata09@gmail.com",
	  "gender" : "M"
	}

Response Success	200
	{
	  "error": false,
	  "message": "Registration successful, please log in."
	}

Response Error	400
	{
	  "error": true,
	  "message": "Email has been used, please log in."
	}
	
	500
	{
	  "error": true,
	  "message": "Internal server error."
	}



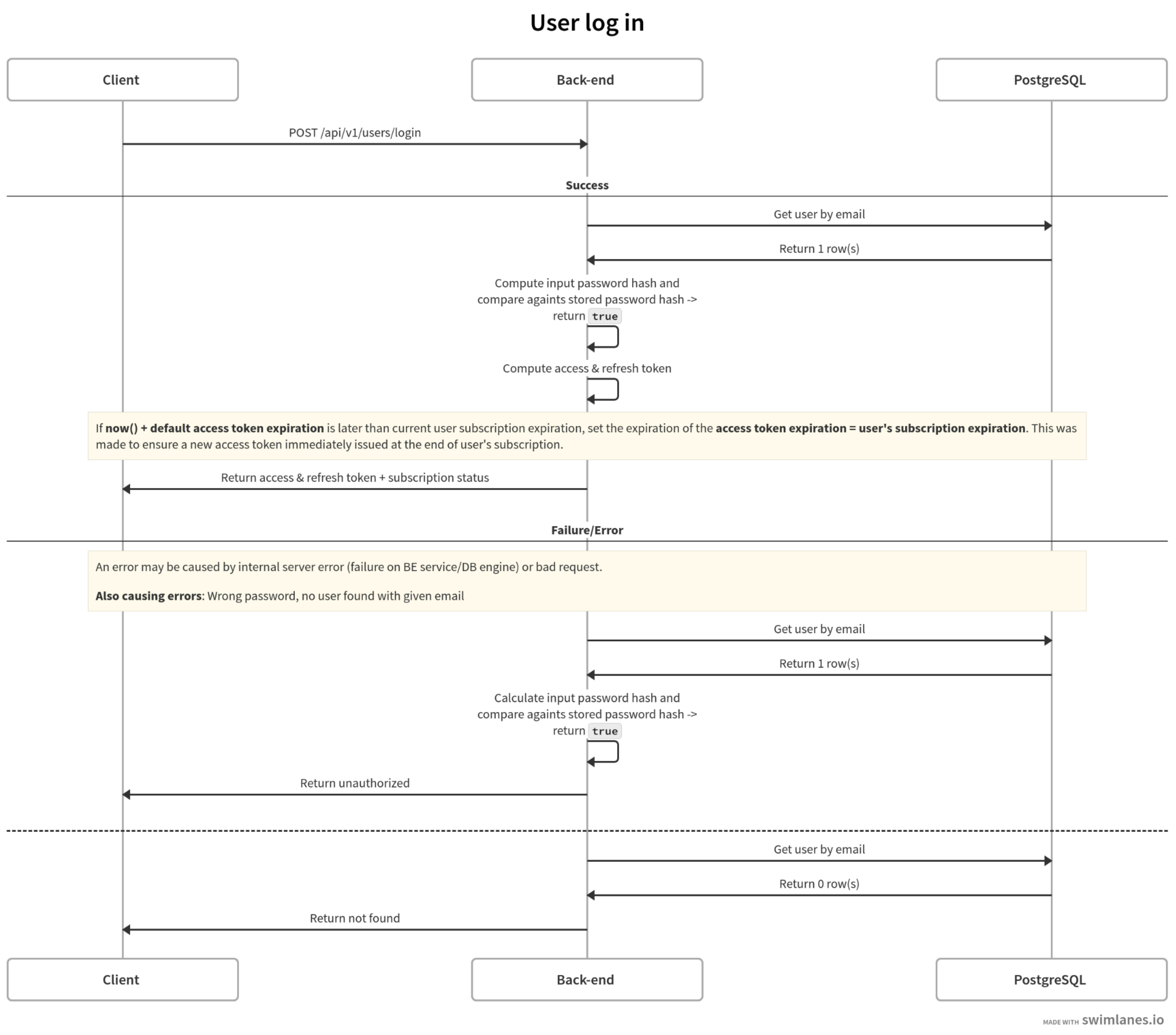
Login

Endpoint	`POST /api/v1/auth/login`

Request body	`email: string required`
	`password: string required`
	`{`
	`  "email": "isimarmata09@gmail.com",`
	`  "password": "v3RYsecrr3t"`
	`}`

Response Success	`200`
	`{`
	`  "error": false,`
	`  "message": "Success",`
	`  "access_token":`
	`"eyJjbGllbnRfaWQiOiJZekV6TUdkb01ISm5PSEJpT0cxaWJEa`
	`HlOVEE9IiwicmVzcG9uc2VfdHlwZSI6ImNvZGUiLCJzY29wZSI`
	`6ImludHJvc2NwZWN0X3Rva2VucywgcmV2b2tlX3Rva2VucyIsI`
	`mlzcyI6ImJqaElSak0xY1hwYWEyMXpkV3RJU25wNmVqbE1iazQ`
	`0YlRsTlpqazNkWEU9Iiwic3ViIjoiWXpFek1HZG9NSEpuT0hCa`
	`U9HMWliRGh5TlRBPSIsImF1ZCI6Imh0dHBzOi8vbG9jYWxob3N`
	`0Ojg0NDMve3RpZH0ve2FpZH0vb2F1dGgyL2F1dGhvcml6ZSIsI`
	`mp0aSI6IjE1MTYyMzkwMjIiLCJleHAiOiIyMDIxLTA1LTE3VDA`

	`3OjA5OjQ4LjAwMCswNTQ1In0", "is_subscriber": false }`
Response Error	`401 { "error": true, "message": "Incorrect password." }` `404 { "error": true, "message": "No user found with given email." }` `500 { "error": true, "message": "Internal server error." }`


User log in
Client
Back-end
PostgreSQL
POST /api/v1/users/login
Success
Get user by email
Return 1 row(s)
Compute input password hash and compare againts stored password hash -> return true
Compute access & refresh token
If **now() + default access token expiration** is later than current user subscription expiration, set the expiration of the **access token expiration = user's subscription expiration**. This was made to ensure a new access token immediately issued at the end of user's subscription.
Return access & refresh token + subscription status
Failure/Error
An error may be caused by internal server error (failure on BE service/DB engine) or bad request.
Also causing errors: Wrong password, no user found with given email
Get user by email
Return 1 row(s)
Calculate input password hash and compare againts stored password hash -> return true
Return unauthorized
Get user by email
Return 0 row(s)
Return not found
Client
Back-end
PostgreSQL
MADE WITH swimlanes.io

Get feeds
@auth_middleware

This API endpoint

<u>For non-subscriber:</u>
Returns list of feed profiles with maximum numbers of **daily activities limit - current User daily activities count.**

<u>For an active subscriber:</u>
Returns 10 profiles

And returns an empty array if there are no more profiles to show.

Endpoint	GET /api/v1/feeds
Headers	Authorization required
Request body	-

Response Success

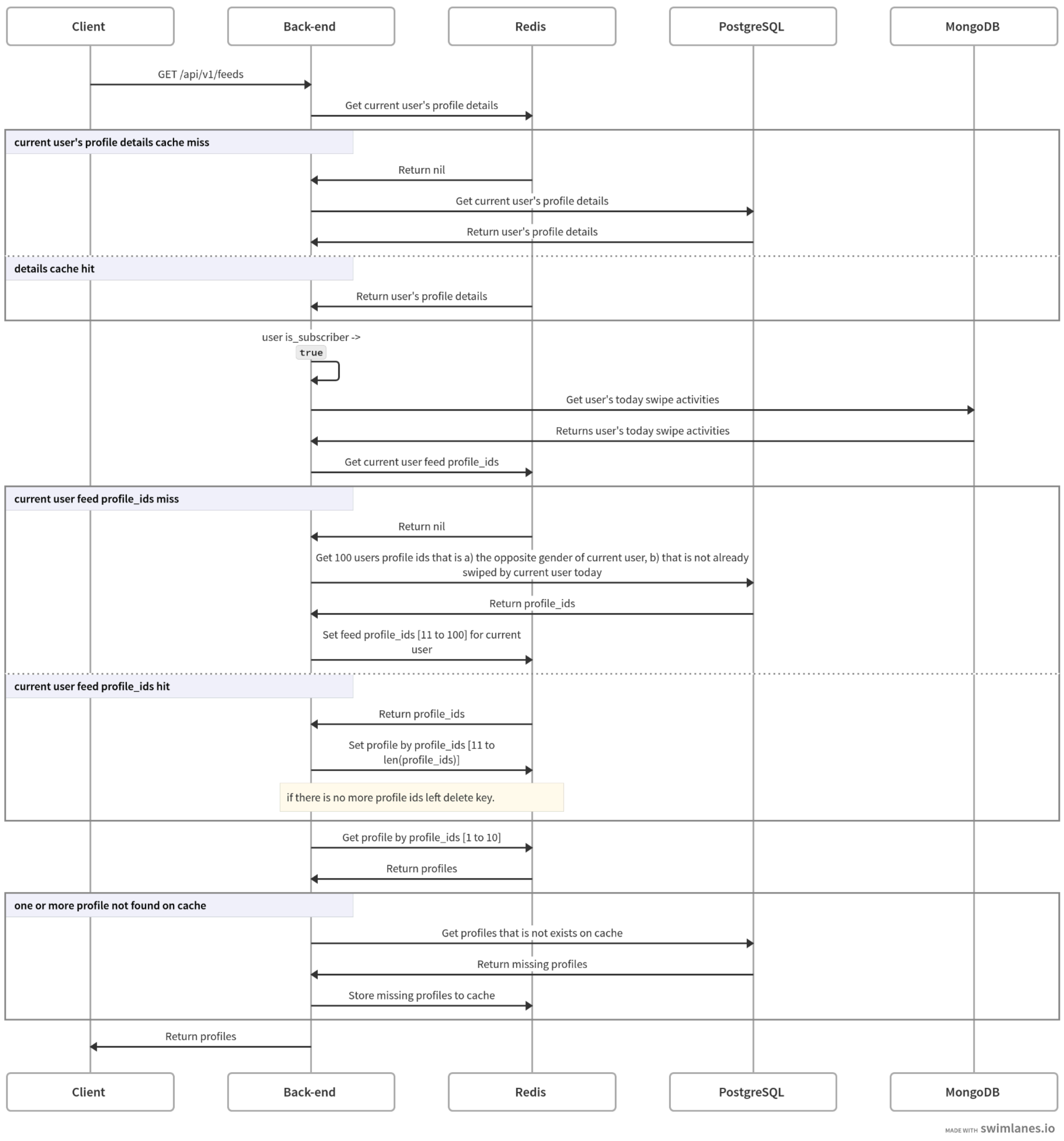
```
200
{
  "error": false,
  "message": "Success.",
  "data": [
    {
      "profile_id":
"b93a20d2-1dab-42cb-88d2-ecfbdd772a98",
      "name": "Siska Alovia",
      "age": 23,
      "zodiac": "Taurus",
      "gender": "F",
      "profile_pic":
"https://img.fixthephoto.com/blog/images/gallery/n
ews_preview_mob_image__preview_11368.png",
      "liked": true
    },
    {
      "profile_id":
"854c7f01-a1c9-474d-b56e-65002d3b7e32",
      "name": "Diana Avirmasi",
      "age": 21,
      "zodiac": "Leo",
      "gender": "F",
      "profile_pic":
"https://media.istockphoto.com/id/1398385367/photo
/happy-millennial-business-woman-in-glasses-posing
-with-hands-folded.jpg?s=612x612&w=0&k=20&c=Wd2vTD
d6tJ5SeEY-aw0WL0bew8TAkyUGVvNQRj3oJFw=",
      "liked": false
    },
    {

    },
    ...
  ]
}
```

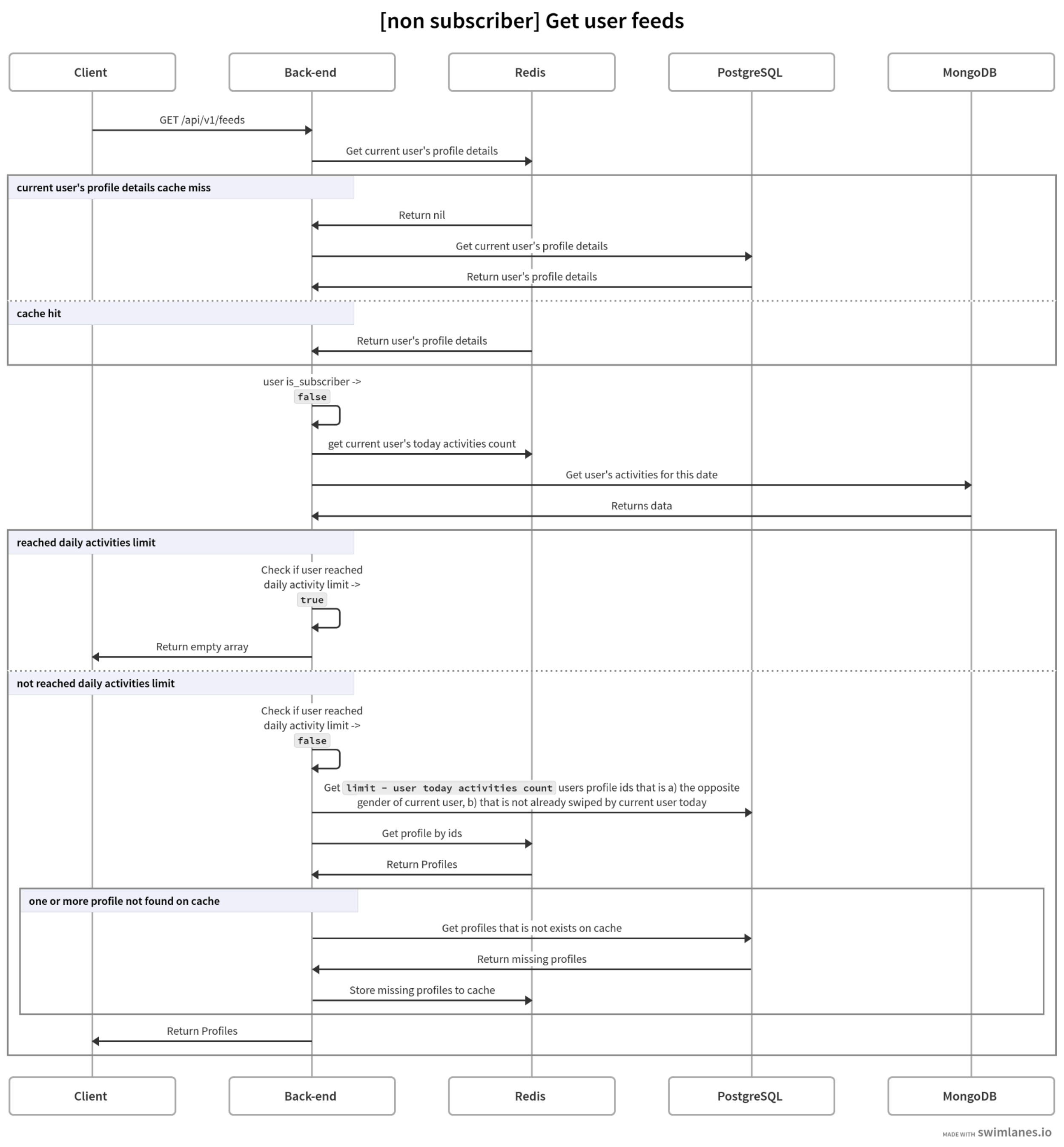
Response Error

```
400
{
```

	"error": true,
	"message": "You’ve reached the daily activities limit."
	}
	
	500
	{
	"error": true,
	"message": "Internal server error."
	}

[subscriber] Get user feeds


Client
Back-end
Redis
PostgreSQL
MongoDB
GET /api/v1/feeds
Get current user's profile details
current user's profile details cache miss
Return nil
Get current user's profile details
Return user's profile details
details cache hit
Return user's profile details
user is_subscriber ->
true
Get user's today swipe activities
Returns user's today swipe activities
Get current user feed profile_ids
current user feed profile_ids miss
Return nil
Get 100 users profile ids that is a) the opposite gender of current user, b) that is not already swiped by current user today
Return profile_ids
Set feed profile_ids [11 to 100] for current user
current user feed profile_ids hit
Return profile_ids
Set profile by profile_ids [11 to len(profile_ids)]
if there is no more profile ids left delete key.
Get profile by profile_ids [1 to 10]
Return profiles
one or more profile not found on cache
Get profiles that is not exists on cache
Return missing profiles
Store missing profiles to cache
Return profiles
Client
Back-end
Redis
PostgreSQL
MongoDB
MADE WITH swimlanes.io


[non subscriber] Get user feeds
Client
Back-end
Redis
PostgreSQL
MongoDB
GET /api/v1/feeds
Get current user's profile details
current user's profile details cache miss
Return nil
Get current user's profile details
Return user's profile details
cache hit
Return user's profile details
user is_subscriber -> false
get current user's today activities count
Get user's activities for this date
Returns data
reached daily activities limit
Check if user reached daily activity limit -> true
Return empty array
not reached daily activities limit
Check if user reached daily activity limit -> false
Get limit - user today activities count users profile ids that is a) the opposite gender of current user, b) that is not already swiped by current user today
Get profile by ids
Return Profiles
one or more profile not found on cache
Get profiles that is not exists on cache
Return missing profiles
Store missing profiles to cache
Return Profiles
MADE WITH swimlanes.io

Skip
@auth_middleware

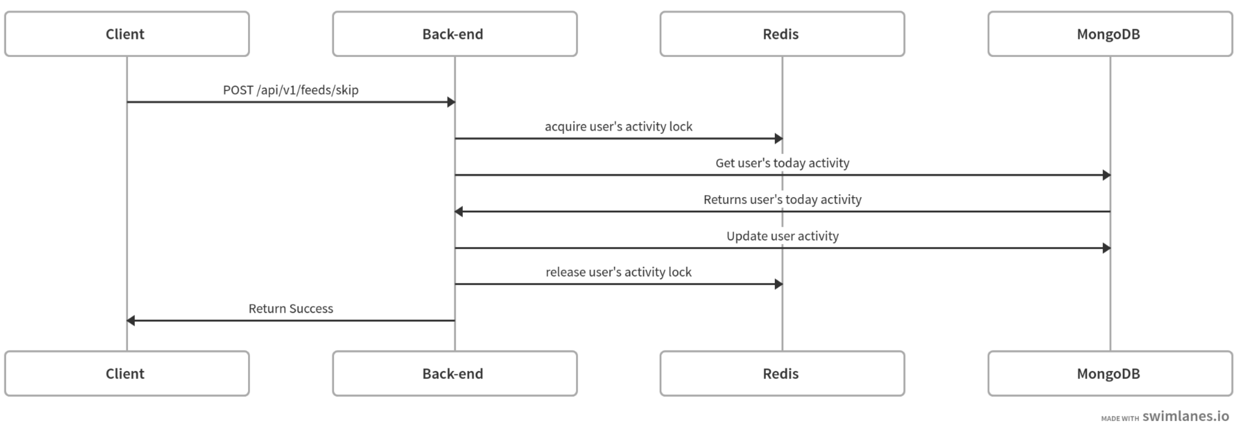
For non-subscriber:
On the client app (mobile app), the 429 HTTP code response should modify the local state of the current user to be not able to see more profiles.

For an active subscriber:

If `profiles[currentProfileIndex + 2]` is none, `Get Feeds` must be called again, to make sure the next profile (if any) is always ready.

Endpoint	`POST /api/v1/feeds/skip`
Headers	`Authorization required`
Request body	`{ "profile_id": "adf6c7d4-aeec-428e-b94b-14519b8346c0" }`
Response Success	`200 { "error": false, "message": "Success." }`
Response Error	`429 { "error": true, "message": "You’ve reached the daily activities limit." }` `500 { "error": true, "message": "Internal server error." }`

Skip a profile



Like
@auth_middleware

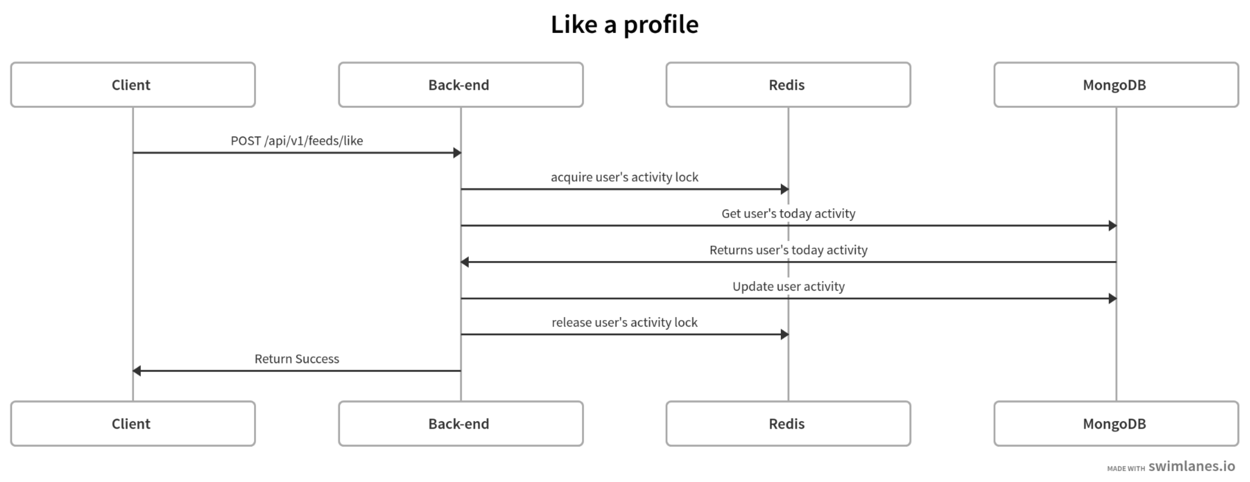
For non-subscriber:
On the client app (mobile app), the 429 HTTP code response should modify the local state of the current user to be not able to see more profiles.

Endpoint	POST /api/v1/feeds/like
Headers	Authorization required

Request body	{
	  "profile_id":
	"a711f350-5d6e-4fb4-933c-d3baa2a6d732"
	}

Response Success	200
	{
	  "error": false,
	  "message": "Success."
	}

Response Error	429
	{
	  "error": true,
	  "message": "You’ve reached the daily activities limit."
	}
	
	500
	{
	  "error": true,
	  "message": "Internal server error."
	}



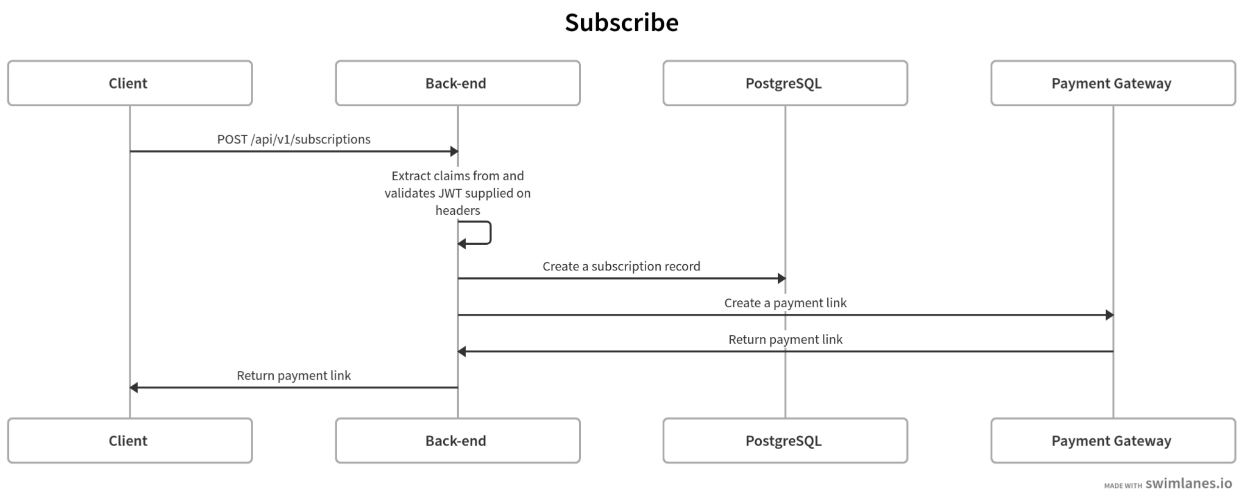
Subscribe
@auth_middleware

Endpoint	POST /api/v1/subscriptions
Headers	Authorization required

Request	{
	  "package_id": "97b2172f-baa2-4b01-8006-2d37aef48c85"
	}

Response Success	200
	{
	  "error": false,
	  "redirect_to": "https://midtrans.co.id/payment-link/0fc76e00-47c4-400d-8933-734a0d2f2e37"
	}

Response Error	400
	{
	  "error": true,
	  "message": "You have an active subscription."
	}
	
	500
	{
	  "error": true,
	  "message": "Internal server error."
	}



4. Non Functional

Locking Mechanism

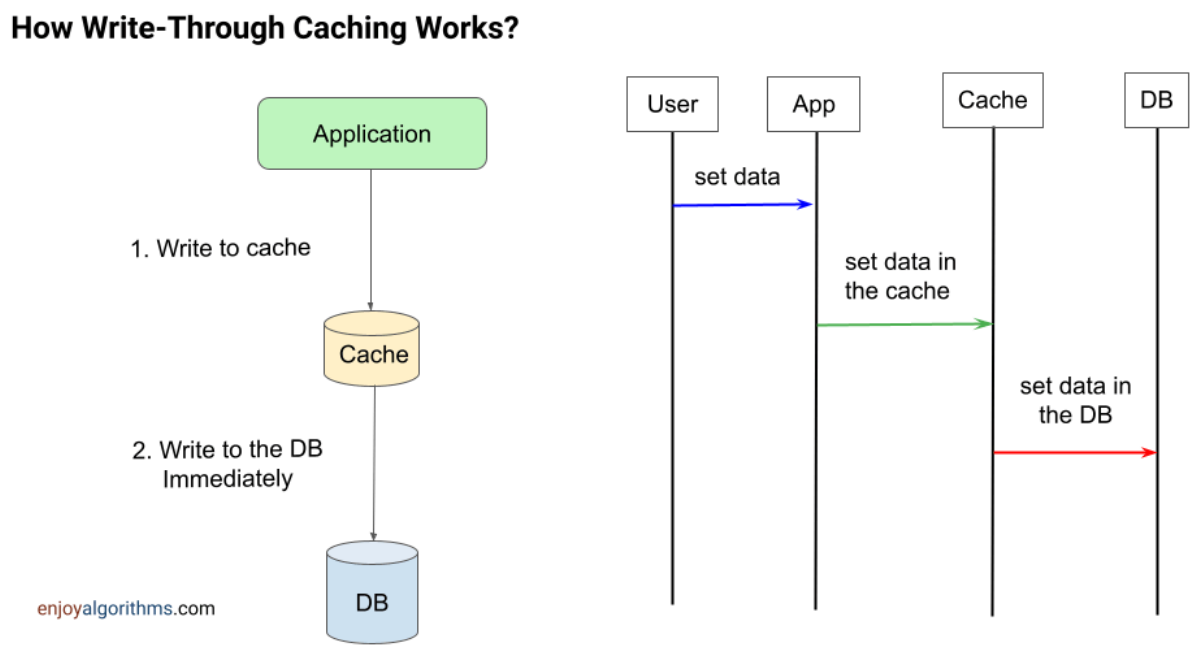
To prevent frequent primary database hits, a caching mechanism was implemented for:

a. Storing users' daily activity count.
b. Storing profile details to show on feeds.

Caching

To avoid a user being able to overrun their daily activity (e.g. a user logged in from two or more devices and doing in-app activity nearly simultaneously) limit, a locking mechanism was implemented on the pass & like function.

Caching strategy: write-through caching



This architecture was selected to ensure the availability of profile data (for feed) on the cache.