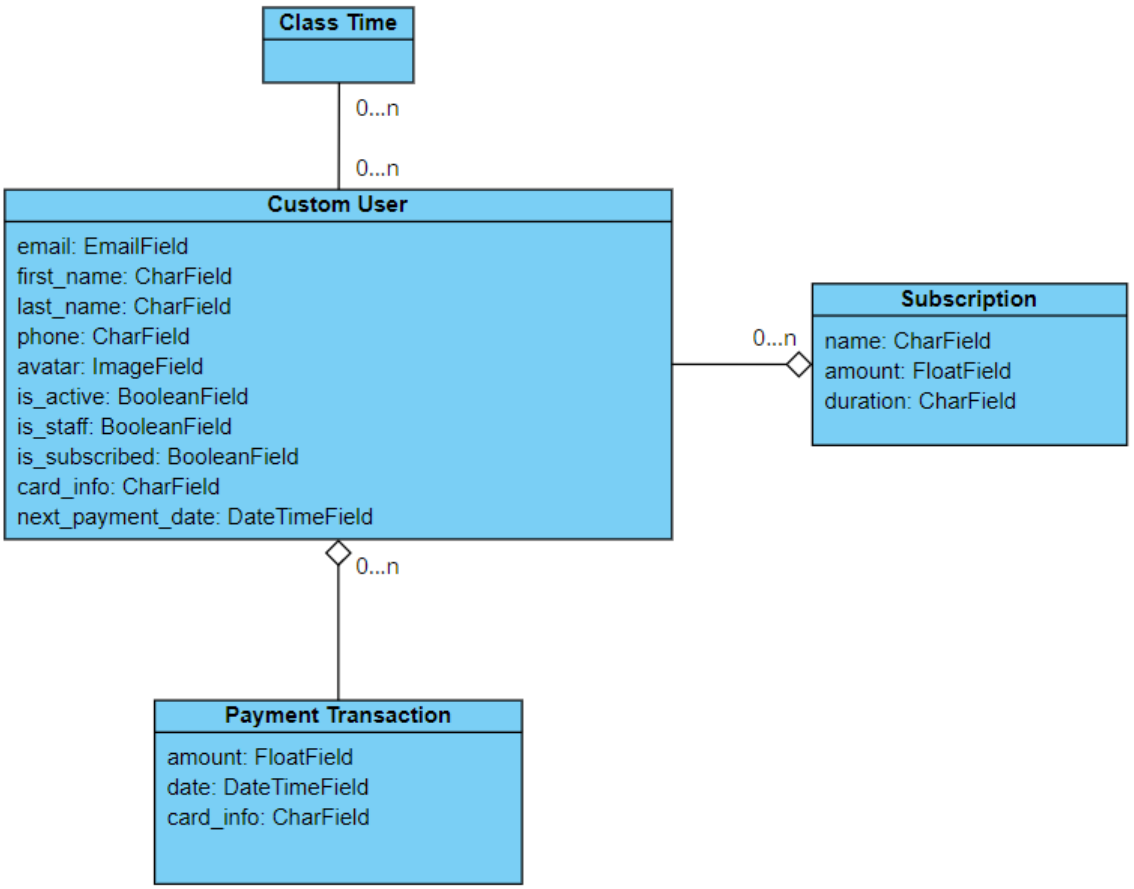
Set up the environment, create a superuser, and run the project:

```
source startup.sh
./PB/manage.py createsuperuser
./run.sh
```

Accounts

User Story: As a user, I can sign up, log in, log out, and edit my profile. Profile information should include first and last name, email, avatar, and phone number.

Implemented by: Dhairya



Create a custom user account with given parameters.

URL: {{base_url}}/accounts/register/

Method: POST

Authorization: None

Notes:

Parameters:

email

first_name

last_name

phone

- input format: ???-???-???? where ? is a digit

avatar

- input format: should be a file

password1

password2

Log in as a custom user if credentials are valid.

URL: {{base_url}}/accounts/login/

Method: POST

Authorization: None

Notes:

Parameters:

email

password

Log out as a custom user that is currently logged in.

URL: {{base_url}}/accounts/logout/

Method: GET

Authorization: None

Notes:

Parameters: None

Change logged in user's password

URL: {{base_url}}/accounts/change_password/

Method: PUT

Authorization: Bearer Token

Notes: This works only if the user is able to recite the old password and provide a valid new password.

Parameters:

old_password

- input format: value of old password

password

password2

Update logged in user's profile with any of the given parameters

URL: {{base_url}}/accounts/update_profile/

Method: PATCH

Authorization: Bearer Token

Notes: Providing only one parameter or multiple parameters both work

Parameters:

email

first_name

last_name

phone

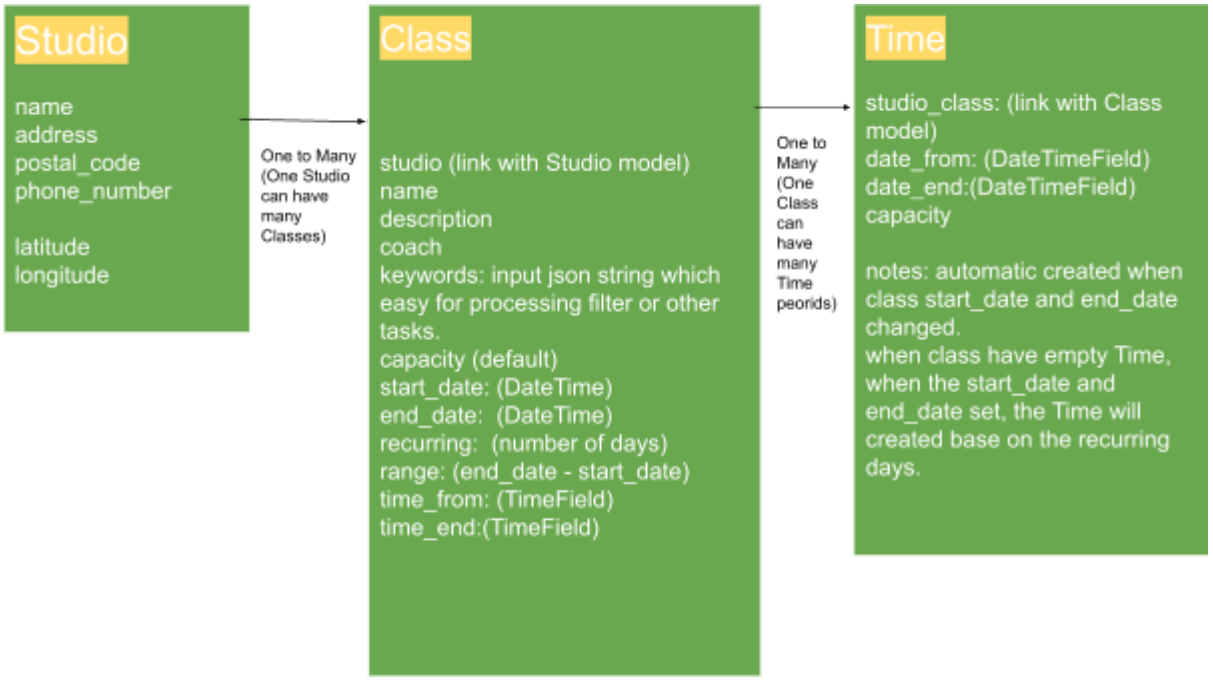
avatar

Studios

User Story: As the **website admin** (i.e., an account that has access to the admin panel), I can create/edit/delete studios. A studio has a name, address, geographical location, postal code, phone number, and a set of images.

As the website admin, I can update the amenities of a studio (type and quantity).

Implemented by: Kehui



User Story: As a user, I want to list studios based on geographical proximity to a specific location (my current location). Studios must list starting from the closest one, each having a drop pin on the map.

User Story (from searching and filtering) : As a user, I want to search/filter through the listed studios (mentioned earlier). Search/filter includes studio name, amenities, class names, and coaches that hold classes in those studios.

Implemented by: Jooyoung

Get studio list

URL: {{base_url}}/studios/list/

Method: GET

Authorization: None

Notes: Params are for searching: if input none all studios are shown.

Studios are ordered by the distance from the user's current location.

Parameters:

name

coach

class

amenity

quantity

User Story: As a user, I can click on each of the studios and move to the studio page. This page should contain the general information of that studio, along with its address, location, and a link to get the directions.
Implemented by: Jooyoung

> **Get Studio Details**
> **URL:** {{base_url}}/studios/detail/{{int: studio_id}}/
> **Method:** GET
> **Authorization:** None
> **Notes:** Shows id, name, address, postalcode, phone number, latitude and longitude of a specific studio. The link for directions will be implemented in frontend.
> **Parameters:** None

Classes

User Story: As the **website admin**, I can create/edit a class in a specific studio. A class has a name, description, coach, a list of keywords (e.g., upper-body, core, etc.), capacity, and times. Times indicate the recurring instances of the class. For example, a HIIT session is held on Mondays from 8:00-9:00 am.
Implemented by: Kehui

User Story: As the **website admin**, I can cancel a specific class; either one instance or all the future occurrences of the class.
Implemented by: Kehui

User Story: As the **website admin**, I can update the amenities of a studio (type and quantity).
Implemented by: Kehui

User Story: As a user, I want to see the class schedule of a specific studio on its page. Classes must appear in the order of their start time (from now), and the class information must be shown. Past or canceled classes should not be listed.

Implemented by: Jooyoung

Get Class Schedules of a specific studio

URL: {{base_url}}/studios/class/list/{{int: studio_id}}/

Method: GET

Authorization: None

Notes: Shows class schedules that have not been passed.
Information of the studio, starting/ending date, and the current capacity are shown.
The schedules are listed in chronological order.

Parameters: None

User Story: As a user, I can enroll/drop a class (either one instance or all future occurrences) that has not started yet and has not reached its capacity. This can only happen if I have an active subscription.

Implemented by: Jooyoung

Enroll in all class instances

URL: {{base_url}}/studios/class/{{int: class_id}}/enroll/

Method: GET

Authorization: Bearer Token

Notes: Automatically skips through: classes that are full in capacity and classes that are passed / already enrolled. Does not display error messages (Just says the user enrolled into all 'available' classes.)

Parameters: None

Enroll in only one class instance

URL: {{base_url}}/studios/class/times/{{int: time_id}}/enroll/

Method: GET

Authorization: Bearer Token

Notes: Checks if the class is full in capacity or if the class is passed / already enrolled. Does give the user a specific error message if the user fails to enroll.

Parameters: None

Drop all class instances

URL: {{base_url}}/studios/class/{{int: class_id}}/drop/

Method: GET

Authorization: Bearer Token

Notes: Automatically skips through classes that are passed / already dropped. Does not display error messages (Just says the user dropped all 'available' classes.)

Parameters: None

Drop only one class instance

URL: {{base_url}}/studios/class/times/{{int: time_id}}/drop/

Method: GET

Authorization: Bearer Token

Notes: Checks if the class is passed / already dropped. Does give the user a specific error message if the user fails to drop.

Parameters: None

User Story: As a user, I want to see my class schedule and history in chronological order.

Implemented by: Jooyoung

Get user's class schedules that are upcoming

URL: {{base_url}}/accounts/time/upcoming/

Method: GET

Authorization: Bearer Token

Notes:

Parameters: None

Get user's class schedules history record

URL: {{base_url}}/accounts/time/history/

Method: GET

Authorization: Bearer Token

Notes:

Parameters: None

Searching and Filtering

User Story: As a user, I want to search/filter a studio's class schedule. The search/filter can be based on the class name, coach name, date, and time range.

Implemented by: Kehui

Search/filter classes

URL:{{base_url}}/studios/class/filter/{{int:studios_id}}/

Method: GET

Authorization: None

Notes: return 404 not found for invalid studios id.

Searching is done by AND conditions

Parameters:

name

coach

start_date

input format: year-month-day (2022-11-17)

end_date

input format: year-month-day (2022-11-17)

range_greater

value: days (for example 100)

description: searching for classes within (end_date - start_date) >= range

range_smaller

value: days (for example 100)

description: searching for classes within (end_date - start_date) <= range

Search class instances

URL:

{{base_url}/studios/class/times/filter/{{int:studios_id}}/

Method: GET

Authorization: None

Notes: return 404 not found for invalid studios id.

Searching is done by OR conditions

Parameters:

name

coach

date_from

input format: year-month-day (2022-11-17)

date_end

input format: year-month-day (2022-11-17)

Subscriptions

Note: Please create a weekly, bi-weekly, monthly and/or yearly gym subscription in the admin panel before testing the subscription apis as they rely on the existence of those subscriptions. The 'name' field is unimportant.

Note 2: The process_payments.py file in PB/accounts is used to create payment transactions for all users that have a next_payment_date of today. If they are still subscribed the file will create a new payment transaction and update required attributes. If they are not subscribed, it will remove the next_payment_date.

This file must be run individually for new payments to be created for users with subscriptions that are due today!

User Story: As the **website admin** I can create/edit/delete the gym subscription plans (e.g., 14.99$ per month or $149.99 per year).
Implemented by: Dhairya

User Story: As a user, I can subscribe to one of the options; the first payment is made immediately after subscription, and each upcoming payment will be automatically made in the beginning of their period.
Implemented by: Dhairya

Subscribe a logged in user to a given membership and charge them first payment

URL: {{base_url}}/accounts/add_subscription/

Method: PUT

Authorization: Bearer Token

Notes:

Parameters:

subscription

- input format: Weekly, Biweekly, Monthly or Yearly

card_info

- input format: 16 digit number

User Story: As a user, I can update my credit/debit card information. All subsequent payments must be made to the updated card.
Implemented by: Dhairya

Update a logged in user's card info

URL: {{base_url}}/accounts/update_card_info/

Method: PUT

Authorization: Bearer Token

Notes:

Parameters:

card_info

- input format: 16 digit number

User Story: As a user, I can see my payment history (amount, card info, date and time, etc.), as well as my future payments.
Implemented by: Dhairya

Get a user's payment history and potential next future payment

URL: {{base_url}}/accounts/payment_history/

Method: GET

Authorization: Bearer Token

Notes: shows payment history **and** next future payment if user is still subscribed (with id=0 for reference) at the **end of the paginated list**

Also, the **recurrence field** is null for all **payment history** just as a placeholder so that the future payment can include its recurrence duration.

Parameters: None

User Story: As a user, I can cancel or update my current subscription. The next payment will be either canceled or updated. In case of cancellation, all class booking after the current billing period will be invalid.

Implemented by: Dhairya

Update user's subscription plan and charge them for the first billing period

URL: {{base_url}}/accounts/update_subscription/

Method: PUT

Authorization: Bearer Token

Notes: If a user has time remaining on their current subscription billing cycle then the new subscription is pre-charged but the next payment date additively includes the current billing cycle and the new prepaid one.

Parameters:

subscription

- input format: Weekly, Biweekly, Monthly or Yearly

Cancel user's subscription

URL: {{base_url}}/accounts/cancel_subscription/

Method: PUT

Authorization: Bearer Token

Notes: Type 'confirm' as the confirm parameter to verify that you want to cancel your subscription.

Parameters:

confirm

- input format: 'confirm'