Design Document

Design of models / API endpoints / Methods and Payloads

General Design Decisions:

1. Auth token is set to 200 days for convenient testing access.
2. The length of the register password should be greater or equal to 8.
3. Login is required for personal view and edit such as profile view, profile edit and etc. Viewing and filtering studios/classes does not require authentication as a new user. You are recommended to Login before you access all endpoints.
4. startup.sh and run.sh should be run in that order for correct operation

List of designed models:
CustomeUser,
Studio, StudioImage, StudioAmenity,
Subscription,
Class, Enrollment, ClassOccurrenceManager, ClassOccurrence
Payment, PaymentManager

Class-based Views: CreateAPIView, RetrieveAPIView, UpdateAIPView,ListAPIView, APIView

APIs:

Accounts

Model:
CustomUser(AbstractUser)
Fields:
username: CharField
first_name: CharField
last_name: CharField
avatar: ImageField
phone_number: Charfield
email: EmailField

Description: Register a user

Endpoint: /accounts/register/

Method: POST,OPTIONS

Payload: email, first_name, last_name, username, phone_number, avatar, password,password2

NOTE: username is unique for each user.

```
SAMPLE:{    "username": "carol8",
            "first_name": "Carol",
            "last_name": "Wan",
             "avatar": null,
             "phone_number": "4168933703",
             "email": "wanchen1999@gmail.com"
            “password”: “123”
            “password2”: “123”                    }
```

Description: Login user

Endpoint: /accounts/login

Method: POST

Payload: username, password,email,phone_number, last_name, first_name,avatar

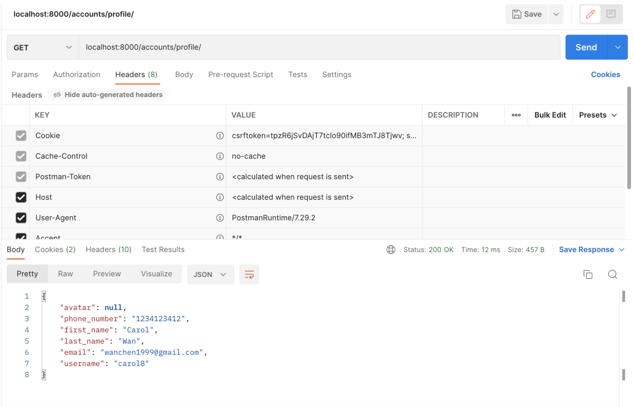
Description: View user’s profile

Endpoint: /accounts/profile/

Method: GET

NOTE: Allow csrftoken in Cookie to successfully login

SAMPLE:



	KEY	VALUE	DESCRIPTION
☑	Cookie	csrftoken=tpzR6jSvDAjT7tclo90ifMB3mTJ8Tjwv; s...	
☑	Cache-Control	no-cache	
☑	Postman-Token	<calculated when request is sent>	
☑	Host	<calculated when request is sent>	
☑	User-Agent	PostmanRuntime/7.29.2	
☑	Accept	*/*	

Status: 200 OK Time: 12 ms Size: 457 B

```
{
    "avatar": null,
    "phone_number": "1234123412",
    "first_name": "Carol",
    "last_name": "Wan",
    "email": "wanchen1999@gmail.com",
    "username": "carol8"
}
```

Description: Edit current user profile

Endpoint: /accounts/profile/edit/

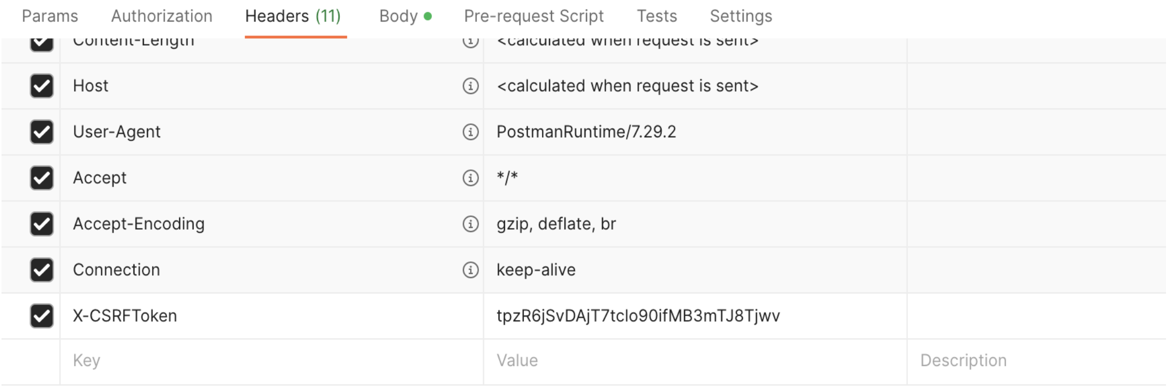
Method: PUT, PATCH

Payload: username, password

Note: Allow csrftoken in Cookie to successfully login

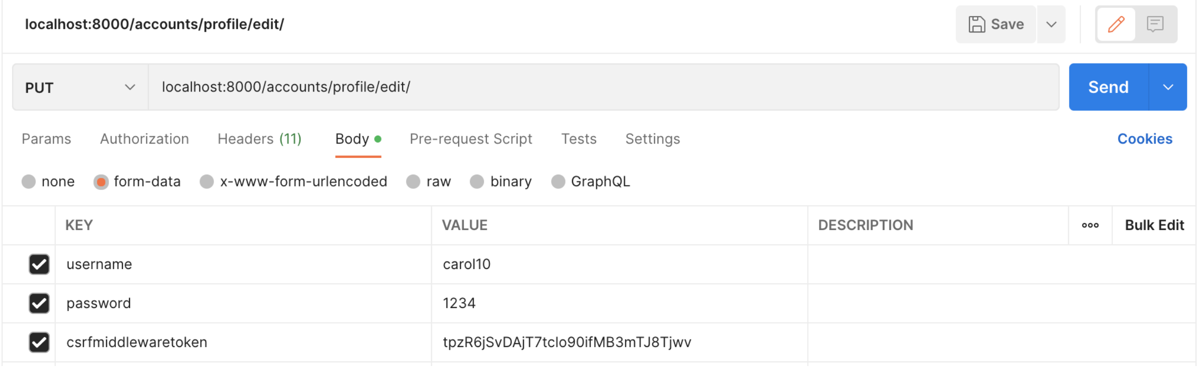
Please find the csrftoken generate in Headers: Cookie

And add { “X-CSRFToken”: generated_token } in Header



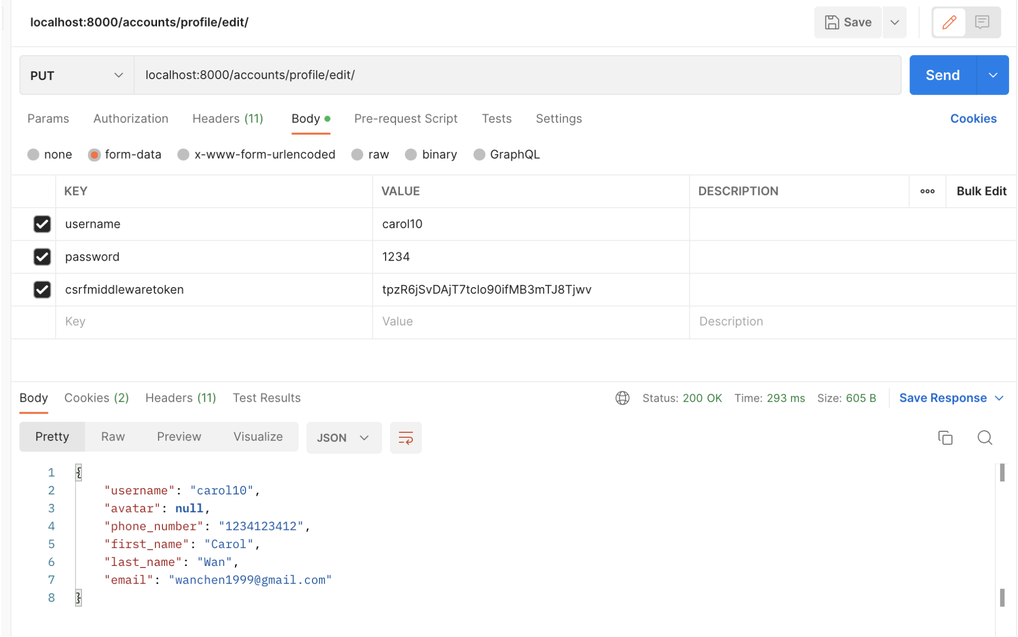
☑	Content-Length	<calculated when request is sent>	
☑	Host	<calculated when request is sent>	
☑	User-Agent	PostmanRuntime/7.29.2	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
☑	X-CSRFToken	tpzR6jSvDAjT7tclo90ifMB3mTJ8Tjwv	
	Key	Value	Description

And add { “csrfmiddlewaretoken” : generated_token } in form-data


localhost:8000/accounts/profile/edit/
Save
PUT
localhost:8000/accounts/profile/edit/
Send
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
Cookies
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL
Bulk Edit

	KEY	VALUE	DESCRIPTION
☑	username	carol10	
☑	password	1234	
☑	csrfmiddlewaretoken	tpzR6jSvDAjT7tclo90ifMB3mTJ8Tjwv	

Here is what postman should look like when you succeed



	KEY	VALUE	DESCRIPTION
☑	username	carol10	
☑	password	1234	
☑	csrfmiddlewaretoken	tpzR6jSvDAjT7tclo90ifMB3mTJ8Tjwv	
	Key	Value	Description

Status: 200 OK Time: 293 ms Size: 605 B

```
{
    "username": "carol10",
    "avatar": null,
    "phone_number": "1234123412",
    "first_name": "Carol",
    "last_name": "Wan",
    "email": "wanchen1999@gmail.com"
}
```

Description: View all classes of current user

Endpoint: /accounts/profile/classes

Method: GET

Description: View all payments of current user

Endpoint: /accounts/profile/payments

Method: `GET`

Studios

Model:
  Studio
Fields:
  name: CharField
  address: CharField
  lon: DecimalField
  lat: DecimalField
  postal_code: CharField
  phone_number: PhoneNumberField

Model:
  StudioImage
Fields:
  image: ImageField
  studio: ForeignKey(Studio)

Model:
  StudioAmenity
Fields:
  name: CharField
  quantity: PositiveIntegerField
  studio: ForeignKey(Studio)

Model:
  Subscription
Fields:
  frequency: CharField
  price: DecimalField
  studio: ForeignKey(Studio)
  subscribers: ManyToManyField(CustomUser)

Model:
  Class
Fields:
  studio: ForeignKey(studio)
  name: CharField
  coach: CharField
  capacity: PositiveIntegerField

description: TextField
start_time: TimeField
end_time: TimeField
recurrences: MultiSelectField
end_date: DateField
keywords: TaggableManager

Model:
ClassOccurrenceManager
Used to create class of specific “date”, “day”, “availability”

Model:
ClassOccurrence
Fields:
date: DateField
day: CharField
available: PositiveIntegerField
cancelled: BooleanField
class_fk: ForeignKey(Class)

Model:
PaymentManager
Fields:
subscriber: ForeignKey(CustomUser)
subscription: ForeignKey(Subscription)
cc_number: CardNumberField
cc_expiry: CardExpiryField
cc_code: SecurityCodeField
date: DateField
amount: DecimalField
paid: BooleanField

- **View all studios**

  Endpoint: /studios/all/

  Method: GET

  Payload: lat, lon

  Description: lat and lon represent the latitude and longitude coordinates. Returns the studios sorted by smallest geographical distance from provided coordinates.

  Notes: lat and lon are optional arguments, if not provided it returns studios sorted by id.

Attention: Please add studios using admin panel
Create superuser before you want to add studio

```
$ python3 manage.py createsuperuser
```

- **Filter studios**

  Endpoint: /studios/filter/

  Method: GET

  Payload: search, ordering, name, amenity, class, coach

  Description: search will do a partial match from any of the following studio fields: name, amenity, class, coach. Ordering will return the studios sorted by the field entered. Valid inputs for this are: id, name, address, lon, lat, postal_code, phone_number. Name, amenity, class, and coach will do a partial match for those specific fields.

- **See studio details**

  Endpoint: /studios/<int:studio_id>/details/

  Method: GET

  Description: returns the details of the specified studio.

- **Subscribe to a studio**

  Endpoint: /<int:studio_id>/subscribe/

  Method: POST, PUT

  Payload: subscription, cc_number, cc_expiry, cc_code

  Description: POST method is for subscribing to a studio or swapping subscription plans. The PUT method is used for updating payment information, which will be applied to future payments.

- **Cancel subscription**

  Endpoint: /<int:studio_id>/subscribe/cancel/

  Method: DELETE

  Description: Cancels the user’s current subscription from the studio

- **View all classes**

  Endpoint: /<int:studio_id>/classes/all/

  Method: GET

  Description: returns all the classes provided by the studio

- **Filter classes**

  Endpoint: /<int:studio_id>/classes/filter/

  Method: GET

  Payload: search, ordering, name, coach, start_time, end_time, start_date, end_date

Description: search will do a partial match from any of the following class fields: name, coach. Ordering will return the studios sorted by the field entered. Valid inputs for this are: name, coach, capacity, start_time, end_time. Name and coach will do a partial match for those specific fields. start_time and end_time will return classes between the specified times, same concept for start_date and end_date.

- **See class details**
  Endpoint: /class/<int:class_id>/details/
  Method: GET
  Description: returns the details of the specified class

- **Enroll in a class**
  Endpoint: /<int:class_id>/enroll/
  Method: GET
  Payload: Occurrence, available_class
  Description: Enroll in the class occurrence defined

- **Drop a class**
  Endpoint: /<int:class_id>/drop/
  Method: POST
  Payload: Occurrence, available_class
  Description: Drops the class occurrence defined