TFC:

The project that represents the backend of our Gym website.

TFC contains five apps: accounts, studios, classes, search, subcriptions. The models, endpoints, methods and payloads are described below.

To generate a token:

Endpoint: 'api/token/'
Method: POST
Required fields: email, password [of a registered user]

Accounts:

For accounts, there is one model

1. GymUser Model:
    a. Includes an user manager to create GymUser instances in admin panel and objects in the model used for creating and using an account
    b. Contains the email, first name, last name, avatar, phone number of the user.
    c. Since this is a gym website, we found it more appropriate to use the email as the username of the client and not have a separate username field. (which may be found in social media websites for example). The rest of the fields are self-explanatory by name.

Accounts Endpoints:

1. Endpoint: accounts/register/
   Method: POST
   Required fields: email, password, first_name, last_name
   Additional fields: avatar, phone_number
   Description: To register an account in the server with the fields

   Note: accounts/register/ is the only endpoint we can run without token authentication, so that we may generate a token with this email and this password.

2. Endpoint: accounts/login/
   Method: POST
   Required fields: email, password [of a registered user]
   Description: To log in to the account

3. Endpoint: accounts/logout/
   Method: POST
   Required fields:

Description: To log out of the account

4. Endpoint: accounts/edit-profile/
   Method: PUT, PATCH
   Required fields: email, password, first_name, last_name [These fields cannot be blank]
   Additional fields: avatar, phone_number
   Description: To edit the account profile with the given fields depending on the request.

Studios:

For studios, there are a total of 3 models.

2. Studio Model:
   a. Includes an object manager to keep track for searching
   b. Sets the name, address, geographical location, postal code, and phone number
3. Image Model:
   a. No way for the user to add multiple images to an object if using ImageField under Studio Model. Created to allow multiple images per studio.
   b. Includes a foreign key for the studio, name of the image, the image, as well as a Boolean to set whether the picture is default.
4. Amenity:
   a. Includes an object manager to keep track for searching
   b. Sets type and quantity

Studio Endpoints:

1. Endpoint: studios/distance/<str:location>/calculate/
   Method: GET
   Description: To calculate the location from the user's input method (current location, pinpoint on the map, or postal code, note that both coordinates as well as postal code is accepted in the parameters) using Google external API
2. Endpoint: studios/<str:studio_id>/<str:location>/information/
   Method: GET
   Description:Returns a list of the studio's information, including a link to get the directions to this location.

Classes:

Classes have total of 3 models,

1. Class model:
   All the basic information about this class, which contain name, description, coach, studio, keyword, capacity, day, time, start_date and end_date
   a. day is the recurring day (Monday, Tuesday…) and time is the 24hr time
2. Recurring class model:
   a. This model contains the id of Class and the date indicate all the future classes.

3. Class User model:
    a. This model contains the id of the recurring classes and user that enrolled into that instance of recurring class.

Classes Endpoints:

1. Endpoint: classes/<studio_id>/all-classes/
   Method: GET
   Display all the classes for such studio
2. Endpoint: classes/<class_id>/recurring-classes/
   Method: GET
   Display all the recurring instances of <class_id>
3. Endpoint: classes/<class_id>/enrol-all/
   Method: POST
   Enroll to all the instances of such class with <class_id> for the current user
4. Endpoint: classes/<recurring_class>/enrol/
   Method: POST
   Enroll to only one instance of class with <recurring_class> for the current user
5. Endpoint: classes/<class_id>/drop-all/
   Method: POST
   Drop all the instance of such class with <class_id> for the current user
6. Endpoint: classes/<recurring_class>/drop/
   Method: POST
   Drop only one instance of class with <recurring_class> for the current user
7. Endpoint: classes/view-classes/
   Method:GET
   Display all the past and futher classes for the current user

   Note: all the enroll/drop views are done by using Class User table to keep track of the attendance in order to check capacity, therefore is the Class User keeps updating instead of Recurring Class.

Search:

Search Endpoints:

1. Endpoint: search/<str:search_by>/<str:search_term>/
   Method: GET
   Description: All searches within the studio will be directed here. The user will be able to choose from four options to search from that would be the parameter search_by: either ‘studio_name’, ‘amenity’, ‘class_name’ or ‘coach_name’. Then they can enter what they want to search that will go into the search_term parameter.
2. Endpoint: search/classes/<str:search_by>/<str:search_term>/
   Method: GET
   Description: All searches for class schedule will be directed here. The user will be able to choose from four options including: ‘class_name’, ‘coach_name’, ‘date’, and ‘time’. They can then put their search term.

Subscriptions:

For subscriptions, there are five models each with their own manager for admin panel views.

1. SubscriptionPlan Model:
    a. Includes a SubcriptionPlanManager to create SubscriptionPlan instances in admin panel and objects in the model used to produce subscription plans.
    b. The model contains plan_number, cost, recurrence.
    c. Plan_number refers to the number of the subscription plan which is set by the admin that creates it.
    d. Cost refers to the cost of the subscription plan.
    e. Recurrence refers to whether the plan charges either every week, month or year.

2. CardInfo Model:
    f. Includes a CardManager to create CardInfo instances in admin panel and objects in the model used for creating credit cards and linking them to the user that provided the information.
    g. The model contains user, cc_number (card number), cc_expiry (card expiry date), cc_code (card security code)
    h. User refers to the user that owns this credit card. The other fields are self-explanatory by name.
    i. To test credit card, please use either of the following as credit card number:
        a. 0000000000000000
        or
        b. 4444444444444444
        Or
        c. Any real card number
3. ActiveSubscription Model:
    j. Includes a SubcriptionManager to create ActiveSubscription instances in admin panel and objects in the model used to link a user who has subscribed to a particular subscription plan using a credit card they provided on a particular day.
    k. The model contains user, subscription_plan, start_date, next_payment_day, end_date, payment_info.
    l. User refers to the user that has subscribed
    m. subscription_plan refers to the subscription plan that user has subscribed to
    n. start_date refers to the date and time of when the user subscribed.
    o. Next_payment_day refers to the day which the user will be charged for renewal of the subscription
    p. end_date refers to the date when the subscription will end (no more renewal)
    q. payment_info refers to the CardInfo model instance that user uses to pay for this subscription
4. Payment model

a. Includes a PaymentManager to create Payment instances in admin panel and objects in the model used to provide a payment history of all payments to the Gym website.
b. The model contains user, active_subscription, cost, payment_datetime, payment_info.
c. User refers to the user that has subscribed
d. active_subscription refers to the subscription that user has.
e. Cost refers to the cost of the subscription renewal.
f. payment_datetime refers to the date and time of when the user paid.
g. payment_info refers to the CardInfo model instance that user uses to pay for this subscription

Subscription Endpoints:

1. Endpoint: subscriptions/subscribe/
   Method: POST
   Required fields: email, plan_number
   Additional fields:
   Description: To subscribe a particular user to a subscription plan with plan_number

2. Endpoint: subscriptions/create-credit-card/
   Method: POST
   Required fields: email, cc_number, cc_expiry, cc_code
   Description: To create a credit card with credentials linked to a user

3. Endpoint: subscriptions/edit-credit-card/
   Method: PUT, PATCH
   Required fields: email, cc_number, cc_expiry, cc_code
   Description: To edit the credit card credentials already existing linked to a user

4. Endpoint subscriptions/payment-history/
   Method: GET
   Required fields: email
   Additional fields:
   Description: To retrieve the payment history of a particular user

5. Endpoint subscriptions/future-payments/
   Method: GET
   Required fields: email
   Additional fields:
   Description: To the get future payment information of a particular user

6. Endpoint: subscriptions/edit-subscription/
   Method: PUT, PATCH
   Required fields: email, plan_number
   Additional fields:
   Description: Switch the users subscription to plan with plan_number

7. Endpoint subscriptions/cancel-subscription/
   Method: PUT, PATCH
   Required fields: email
   Additional fields
   Description: Cancel the users active subscription. Does not let the user use the gym right after.

8. Endpoint subscriptions/record-payments/
   Method: PUT, PATCH
   Required fields:
   Additional fields:
   Description: The server will use a crontab to daily charge all users who have a payment pending on that day, by running a script makepayments.sh which calls this endpoint. This results in recording of transactions. [Essentially only meant for server use]