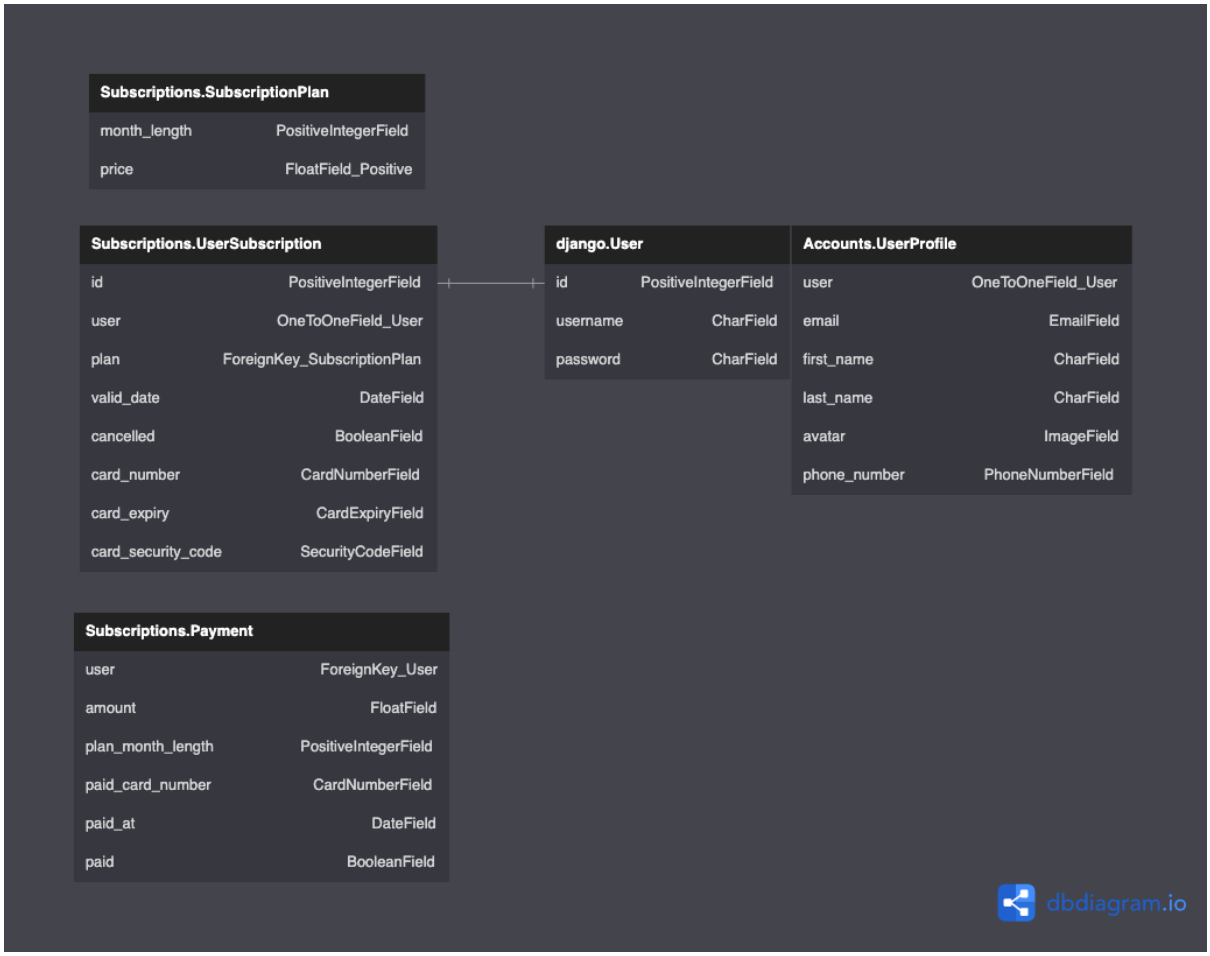
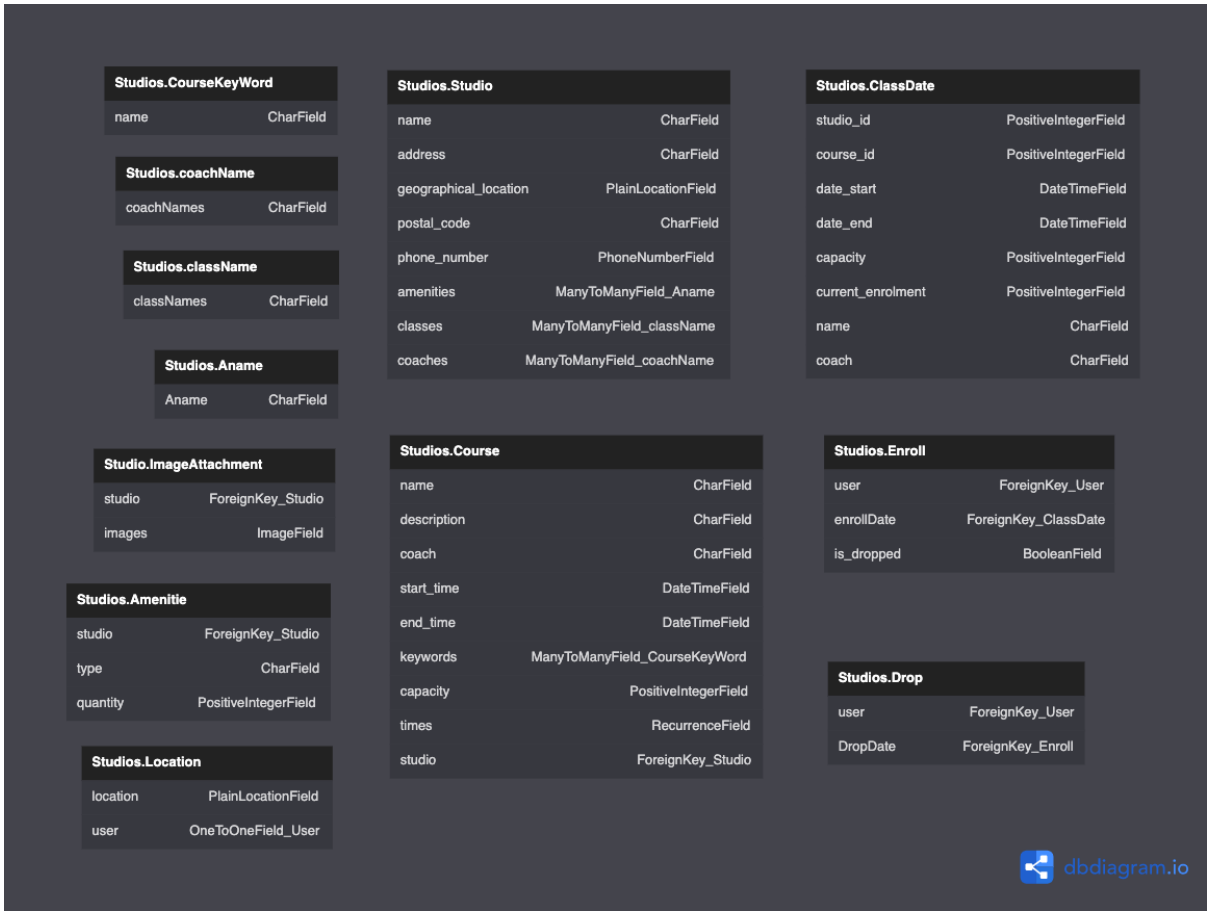
CSC309 API Documentation

Design of Models





- User
  - Django default User
- UserProfile
  - Profile info associated with each user
- SubscriptionPlan
  - The available subscription plan for users, which can only be created by admin
- UserSubscription
  - Subscription info associated with each user
- Payment
  - The paid or future payment
- coachName
  - Store all the coach names and make them unique in the model automatically based on the adding function in the Course model
  - In order to meet the filter/search functionality requirement
- className
  - Store all the class names and make them unique in the model automatically based on the adding function in the Course model
  - In order to meet the filter/search functionality requirement
- Aname

    - Store all the amenity name and make them unique in the mode automatically based on the adding function in the Amenitie model
    - In order to meet the filter/search functionality requirement
- Studio
    - Store required information about studios
    - Add manytomany fields in order to meet the filter/search functionality requirement
- ImageAttachment
    - Store the uploaded images with a foreign key linked to the studio
- Amenitie
    - Admin user will add amenities here in this model
    - Store the amenity type and quantity with a foreign key linked to the studio
- Location
    - ‘Specific location’ mentioned in the third point of Studios part
    - Store with a onetoone field linked with user
- CourseKeyWord
    - Admin user will add course key words here in this model
    - can select the need ones in the ‘keywords’ field in Course model
- ClassDate
    - Automatically created based on the ‘time’ recurrence field in Course model
- Enroll
    - Store all the class dates that has being enrolled by a specific user
    - has a ‘is_dropped’ flag to keep track whether the class date has been dropped
- Drop
    - Store all the class dates that has being dropped by a specific user
    - DropDate must be the dates that has being enrolled

API Endpoints

▼ Note

- Authentication == True
    - This indicates for this endpoint, JWT token is required
- There are a couple of fields from external library. The sample inputs are provided here
    - PhoneField
        - +14379866653
    - CardNumberField
        - 4111111111111111

        - 30569309025904
    - CardExpiryField
        - 2025-02-01
        - 2026-09-02
    - SecurityCodeField
        - 792
        - 1111
        - https://github.com/dldevinc/django-credit-cards/blob/master/tests/test_forms.py
    - PlainLocationField
        - 44.00, 45.00

Accounts

▼ `POST` `/accounts/register/`

- Description
    - Register the username with password.
    - Some additional profile info can be register together as well.
- Payload
    - Required
        - `username`
            - char field
        - `password`
            - char field
        - `password2`
            - char field
            - Description
                - The repeated password
            - Extra validation
                - password == password2 is checked
    - Optional
        - `email`
            - email field
        - `first_name`
            - char field
        - `last_name`
            - char field
- Response
    - Code: 201 if success

    - Code: 400 if bad request
- Note
    - `UserProfile` Model is created when `User` is created

▼ `POST` `/accounts/login/`

- Description
    - Take a user credential and returns an access and refresh token to prove the authentication of the credential
- Payload
    - Required
        - `username`
            - char field
            - Description
                - the username created by by `POST` `/accounts/register/`
        - `password`
            - char field
            - Description
                - the password of the user created by `POST` `/accounts/register/`
- Response
    - Code: 201 if success
    - Code: 400 if fail
    - Code: 401 if unauthorized

▼ `POST` `/accounts/logout/`

- Description
    - Blacklist the refresh token for the user.
    - Authentication: True
- Payload
    - `refresh_token`
        - char field
        - required
- Response
    - Code: 205 if success
    - Code: 400 if bad request (invalid token type or invalid token)
    - Code: 401 if unauthorized

▼ `PUT` `/accounts/{id}/change_password/`

- Description
  - Change the password for the user
  - Authentication: True
- Payload
  - Required
    - `old_password`
      - char field
      - Extra validation
        - Check whether it's the same as the original password.
    - `password`
      - char field
    - `password1`
      - char field
      - Extra validation
        - Check whether it's the same as `password`
- Response
  - Code: 200 if success
  - Code: 400 if bad request
  - Code: 401 if unauthorized

▼ `PUT` `/accounts/update_profile/`

- Description
  - Update the user profile
  - Authentication: True
- Payload
  - Optional
    - [redacted]
      - char field
    - `last_name`
      - char field
    - `email`
      - email field
    - `phone_number`
      - phone field
    - `avatar`
      - image field
      - According to TA, this couldn't be tested without front-end.

- Response
  - Code: 200 if success
  - Code: 400 if bad request
  - Code: 401 if unauthorized

▼ `GET` `/accounts/view_profile/`

- Description
  - View the user profile
  - Authentication: True
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

Subscriptions

▼ `GET` `/subscriptions/all_plans/`

- Description
  - View all subscription plans created by admin panel
- Response
  - Code: 200 if success
  - Code: 400 if bad request

▼ `POST` `/subscriptions/subscribe/`

- Description
  - Subscribe a particular plan for the user
    1. `UserSubscription` is created
       - `plan` should be the `SubscriptionPlan` created by the superuser
       - `user` should be the user
       - `id` should be the same as the user id
       - `valid_date` is set to today + the length of plan (by month)
         - It should be updated whenever a future payment is paid.
       - `cancelled` is set to False
       - all other card info are from the payload
    2. The first payment `Payment` is created
       - `user` should be the user

        - `amount, paid_month_length, paid_card_number` should be from `UserSubscription` or `UserSubscription.plan`
        - `paid_at` should be the date of today.
        - `paid = True`
      3. One future payment `Payment` is created
        - All other fields are the same as above.
        - If we update credit card info or plan, this future payment's amount / paid_month_length / paid_card_number needs to be changed
        - `paid_at` should be the `UserSubscription.valid_date`
        - `paid = False`
    - Authentication: True
- Payload
    - Required
      - `plan_code`
        - positive integer field
        - This is the id of the plan created by the admin.
      - `card_number`
        - card number field
      - `card_expiry`
        - card expiry field
      - `card_security_code`
        - card security code field
- Response
    - Code: 201 if success
    - Code: 401 if unauthorized
    - Code: 400 if fail
- Note
    - if the post has already been called by this user, the subscription won't be changed.
    - If the user wants to change the subscription or card info, need
      - `PUT` `/subscriptions/update_card_info/`
      - `PUT` `/subscriptions/update_plan/`

▼ `PUT` `/subscriptions/cancel_plan/`

- Description
    - Cancel the user subscription plan.
      1. Update the `cancelled` status by payload in `UserSubscription`
        - In payload, `cancelled` is a boolean field, if it's false then nothing will be changed. It should only input True.

2. Delete the future `Payment` from the database.

- Authentication: True

- Payload
  - Required
    - `cancelled`
      - boolean field
- Response
  - Code: 201 if success
  - Code: 401 if unauthorized

Make a payment based on a given date

run `python payment_utils.py --year {int} --month {int} --day {int} --make_payment`

- Get all future `Payment` on this date,
  - for each future payment, if the `cancelled` in `UserSubscription` is not false (plan is not cancelled)
    - Change the `paid` in future `Payment` to True
    - Update the `valid_date` in `UserSubscription` by month length of plan
    - Create a new future `Payment`

Change the valid date for user profile

run `python payment_utils.py --year {int} --month {int} --day {int} --username {str}`

- Change the user's valid date based on the username

▼ `PUT` `/subscriptions/update_plan/`

- Description
  - Update the user subscription plan.
    1. Update the `plan` by payload's plan_code in `UserSubscription`
    2. Change the `amount` and `paid_month_length` in future `Payment`
  - Authentication: True
- Payload
  - Required
    - `plan_code`
      - integer
- Response
  - Code: 201 if success
  - Code: 401 if unauthorized

▼ `PUT` `/subscriptions/update_card_info/`

- Description
  - Update the card information for user subscription profile
    1. Update the card info by payload in `UserSubscription`
    2. Change the `paid_card_number` in future `Payment`
  - Authentication: True
- Payload
  - Required
    - `card_number`
      - card number field
    - `card_expiry`
      - card expiry field
    - `card_security_code`
      - card security code field
- Response
  - Code: 201 if success
  - Code: 401 if unauthorized

▼ `GET` `/subscriptions/payment_history/`

- Description
  - Get all `Payment` belonging to this user
  - Authentication: True
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `PUT` `/subscriptions/reactivate_plan/`

- Description
  - Reactivate the cancelled subscription
    1. Update the `plan` , `cancelled` and all card info in `UserSubscription`
       - In payload, `cancelled` is a boolean field, if it's True then nothing will be changed. It should only input False.
    2. Make the initial `Payment` if the `valid_date` has passed.
    3. Create the future `Payment`
  - Authentication: True
- Payload
  - Required

        - `cancelled`
            - boolean field
        - `plan_code`
            - positive integer field
        - `card_number`
            - card number field
        - `card_expiry`
            - card expiry field
        - `card_security_code`
            - card security code field
    - Response
        - Code: 201 if success
        - Code: 401 if unauthorized

▼ `GET` `/subscriptions/view_subscripition/`

- Description
    - Get the `UserSubscription` profile belonging to this user
    - Authentication: True
- Response
    - Code: 200 if success
    - Code: 401 if unauthorized

Studios, Classes and Search/Filter

Authentication: True for all endpoints here

▼ `POST` `/studios/create_location/`

- Description
    - Create a location for the user based on the input
- Payload
    - location
        - PlainLocationField
        - required
- Response
    - Code: 200 if success
    - Code: 401 if unauthorized

▼ `PUT` `/studios/update_location/`

- Description
  - Update a location for the user based on the input
- Payload
  - location
    - PlainLocationField
    - required
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `GET` `/studios/list_studios/`

- Description
  - List all studios that are created by the admin and sorted by geographical proximity to the user's location
  - **Need to create a location for the user by `POST` `/studios/create_location/` before using it.**
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `GET` `/studios/view_studio/{id}/`

- Description
  - View the studio profile by studio id
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `GET` `/studios/list_classes/studio/{id}/`

- Description
  - List the classes at a particular studio, and can also perform filter and search by input parameters
- Parameters
  - name
  - coach (name)
  - search (string)
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `POST` `/studios/enroll_classdate/`

- Description
  - enroll one class date (i.e. recurring class) created by admin panel for the user
- Payload
  - enrollDate
    - positive integer field
    - the id of class date to enroll
- Response
  - Response
    - Code: 201 if success
    - Code: 401 if unauthorized
    - Code: 400 if fail

▼ `POST` `/studios/enroll_class/`

- Description
  - enroll all class dates (i.e. recurring class) for a class created by admin panel for the user
- Payload
  - course_code
    - positive integer field
    - the id of class (course) to enroll
- Response
  - Response
    - Code: 201 if success
    - Code: 401 if unauthorized
    - Code: 400 if fail

▼ `GET` `/studios/list_enrolled_classdate/`

- Description
  - List all classes (class schedule) - all future class dates that have been enrolled by the user
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `POST` `/studios/drop_classdate/`

- Description
  - drop one class date (i.e. recurring class) that has been enrolled by the user

- Payload
  - dropDate
    - positive integer field
    - the id of class date to enroll
- Response
  - Response
    - Code: 201 if success
    - Code: 401 if unauthorized
    - Code: 400 if fail

▼ `POST` `/studios/drop_class/`

- Description
  - drop all class dates (i.e. recurring class) for a class that has been enrolled by the user
- Payload
  - course_code
    - positive integer field
    - the id of class (course) to drop
- Response
  - Response
    - Code: 201 if success
    - Code: 401 if unauthorized
    - Code: 400 if fail

▼ `GET` `/studios/history/`

- Description
  - List the enrollment history of the user (all the past enrolled class dates based on the date now)
- Response
  - Code: 200 if success
  - Code: 401 if unauthorized

▼ `GET` `/studios/list_studios_search/`

- Description
  - Search and filter the studios by provided parameters
- Parameters
  - Optional
    - name
    - amentities (id)

        - classes (id)
        - coaches (id)
        - search (string)
    - Response
        - Code: 200 if success
        - Code: 401 if unauthorized

▼ `GET` `/studios/list_classes/studio/{id}/search_date/{filterDate}/`

- Description
    - filter the class at a specific studio by date (XXXX-XX-XX)
- Response
    - Code: 200 if success
    - Code: 401 if unauthorized

▼ `GET` `/studios/list_classes/studio/{id}/search_time/{filterstart}/{filterend}/`

- Description
    - filter the class at a specific studio by time range (xx:xx:xx)
- Response
    - Code: 200 if success
    - Code: 401 if unauthorized