STUDIO/CLASS/AMENITY

Users:

accounts/register

- Register a new user account
- Receive POST request to API for checking register endpoint. Fields are username, password, password2, email, first_name, last_name, avatar and phone number should be added to the body. If fields passed validations(password matches), the new user detail will be returned.

accounts/api/token

- Send POST request to API. Request body must have two parts : username and password.
- Returns a token valides for 1 day

accounts/api/token/refresh

- Send POST request to API. Request body must have refresh token
- Returns a token valides for 1 day

accounts/update_profile/<str:username>

- Send PATCH or PUT request to API. Request body must have an auth token, first and last name, avatar and phone number should be added to the body(optional if using PATCH).
- Return the user detail after change

studios/[studio id]/view

- Get information about a specific studio studios/[studio id]/view/mylocation=[longitude],[latitude]

- Get information about a specific studio with a link to directions

studios/search

- Search for a studio through the URL
- Supports name, fitnessclass, amenity, coach (all strings)
- Example search: studios/search?name=[studio name]&amenity=[amenity type]
- Paginated with PageNumberPagination (append &page=[pagenum] to search query)

studios/list/mylocation=[longtitude],[latitude]

- Get a list of the studios closest to the provided coordinates
- User must provide location via latitude and longitude
- Paginated with PageNumberPagination (append /?=page=[pagenum])

studios/[studio id]/schedule

- Get the class schedule of a specific studio
- Paginated with PageNumberPagination (append /?=page=[pagenum])

studios/class/[class id]/enroll_[mode]

- Takes “one” or “al”l for enrolling in 1 class or all recurring classes
- Takes GET request

studios/class/[class id]/drop_[mode]

- Same usage as enroll

studios/class/schedule

- Takes GET request and returns the user’s upcoming classes

studios/class/history

- Same as my_schedule but for past classes

studios/[studio id]/class/list

- Get a list of classes from a specific studio
- Paginated with PageNumberPagination (append /?=page=[pagenum])

studios/class/[class_id]/view

- Get information about a specific class

studios/class/search

- Search for a class through the URL
- Supports name, coach, date, and time_range
- Name and coach are strings, date follows format YYYY-MM-DD, time_range must be 24 hour time and follows the format HH:MM-HH:MM ([start time]-[end time])
- Example search: studios/class/search?name=[class_name]&time_range=11:00-13:00
- Paginated with PageNumberPagination (append &page=[pagenum] to search query)

card/view/

- This route is available to authenticated users through TokenAuth.
- Retrieve the saved credit card information of an user on a GET request. The user must be registered through the Account application.
- Example: GET hostname/payment/card/view/
- Example response:

```
{
  "data": {
    "id": 7,
    "user_id": 1,
    "card_number": 1234123412341234,
    "card_expiration_date": "2023-05-13",
    "card_holder_firstname": "Harry",
    "card_holder_lastname": "Wang"
  }
}
```

card/add/

- This route is available to authenticated users through TokenAuth.
- Add credit card information to the database on a POST request with JSON payload, The user must be registered in the system. If there is already a card registered, there will be no effect. Error checking will be performed to ensure the payload is valid.
- Example: POST hostname/payment/card/add/
    - Payload:

```
{
  "data": {
    "card_number": "1234123412341234",
    "card_expiration_date": "2023-05-13",
    "card_holder_firstname": "Harry",
    "card_holder_lastname": "Wang"
  }
}
```

card/edit/

- This route is available to authenticated users through TokenAuth.
- Update the credit card information registered by the user on a PUT request with JSON payload. The user must be registered in the system with a card that has already been registered. Error checking will be performed to ensure the payload is valid. An 400 error will be returned if any error occurs.
- Example: PUT hostname://payment/card/edit/
    - Payload:

```
{
  "data": {
    "card_number": "1234123412341234",
    "card_expiration_date": "2023-05-13",
    "card_holder_firstname": "Harry",
    "card_holder_lastname": "Wang"
  }
```

```
            }
```

- Example response:

```
{
  "success": "Card updated successfully"
}
```

subscription/plans/all/

- This route is available to both anonymous and authenticated users.
- Returns all of the subscription plans that is available via JSON response on a GET request.
- Example: GET hostname/payment/subscription/plans/all/
- Example response:

```
{
  "data": [
    {
      "id": 1,
      "name": "GymPro",
      "description": "Access to yoga, gym, and acrbic room.",
      "price": 30.0,
      "is_live": true,
      "is_monthly": true
    },
    {
      "id": 2,
      "name": "Gym Elite",
      "description": "Access to gym, yoga, aerobic room, and pool.",
      "price": 40.0,
      "is_live": true,
      "is_monthly": true
    }
  ]
}
```

subscription/view/

- This route is available to authenticated users through TokenAuth.
- Returns the current subscription of an authenticated user on a GET request. The user must be registered in the system.

- Example: GET hostname/payment/subscription/view/
- Example response:

```
{
  "data": {
    "id": 24,
    "user_id": 1,
    "subscription_plan_id": 1,
    "date_time": "2022-11-17T18:58:04.070043Z"
  }
}
```

subscription/subscribe/

- This route is available to authenticated users through TokenAuth.
- Subscribe to a subscription plan on a POST request. The user that makes the request must have a credit card registered in the system. The selected subscription plan should also be a valid plan_id that is currently open for subscription. Error checking will be performed to ensure all above requirements. 400-level errors will be returned on broken premises.
- Example: POST hostname/payment/subscription/subscribe/
    - Payload:

      ```
      {"data": { "subscription_plan_id": "1" }}
      ```
- Example response:

```
{
  "success": "Successfully subscribed to Goofy Plan."
}
```

subscription/edit/

- This route is available to authenticated users through TokenAuth.
- Update the subscription plan on a PUT request. The user must have a credit card registered in the system. When making an update, the current subscription will be terminated immediately, and a new subscription with the new subscription plan will be added to the database. In the meantime, an payment will be made, and recurring payment set.

- Error checking will be performed to ensure all requirements and enforce valid payload. 400-level errors will be returned on broken premises.
- Example: PUT hostname/payment/subscription/edit/
    - Payload:

```
{
  "data": {
    "subscription_plan_id": "3"
  }
}
```

subscription/cancel/

- This route is available to authenticated users through TokenAuth.
- Cancel the current subscription on a DELETE request. The user must have a valid subscription in the system. 400-level errors will be returned on broken requirements.
- Upon cancellation, any unpaid recurring payment for this subscription will be removed for this user.
- Example: DELETE hostname//payment/subscription/cancel/

payment/upcoming/

- This route is available to authenticated users through TokenAuth.
- Returns all of the upcoming payment records of the user on a GET request. The user must be registered and authenticated.
- Example: GET hostname/payment/upcoming/
- Example Respons:

```
{
  "data": {
    "id": 33,
    "user_id": 1,
    "subscription_plan_id": 1,
    "amount": 20.0,
    "date_time": "2022-12-17T18:58:04.081392Z",
    "card_number": "1234123412341234",
    "card_expiration_date": "2023-05-13",
    "card_holder_firstname": "Harry",
    "card_holder_lastname": "Wang",
    "is_paid": false
  }
}
```

payment/history/

- This route is available to authenticated users through TokenAuth.
- Returns all of the past payment records of the user on a GET request. The user must be registered and authenticated.
- Example: GET hostname/payment/history/
- Example response:

```
{
  "data": [
    {
      "id": 28,
      "user_id": 1,
      "subscription_plan_id": 1,
      "amount": 20.0,
      "date_time": "2022-11-16T22:16:06.852421Z",
      "card_number": "1234123412341234",
      "card_expiration_date": "2023-05-13",
      "card_holder_firstname": "Harry",
      "card_holder_lastname": "Wang",
      "is_paid": true
    },
    {
      "id": 30,
      "user_id": 1,
      "subscription_plan_id": 3,
      "amount": 40.0,
      "date_time": "2022-11-16T22:16:21.996982Z",
      "card_number": "1234123412341234",
      "card_expiration_date": "2023-05-13",
      "card_holder_firstname": "Harry",
      "card_holder_lastname": "Wang",
      "is_paid": true
    }
  ]
}
```

Admins:

(moved to admin panel instead.. Didn't realize they weren't supposed to be APIs)

studios/create

- Creates a studio from a POST request
- Takes name (TextField), address (TextField), locationX (FloatField), locationY (FloatField), postalCode (TextField), phoneNumber (TextField), and images (ImageField)(imagefield accessible through browser or admin panel)

studios/[studio id]/edit

- Edits information for a specific studio from PUT or PATCH request
- Fields same as create

studios/[studio id]/delete

- Takes a DELETE request and deletes that studio (will cascade to FitnessClass and Amenity models)

studios/[studio id]/amenities/create

- Create amenities for a specific studio
- Takes type (TextField), quantity (PositiveIntegerField)
- Amenity type and studio are a unique pair (cannot have 2 instances of the same amenity type in the same studio)

studios/[studio_id]/amenities/list

- Returns list of amenities of a specific studio
- Paginated with PageNumberPagination (append /?=page=[pagenum])

studios/amenities/edit/[amenity id]

- Send a PUT or PATCH request to update the amenity quantity (use 0 if that amenity is no longer available) studios/[studio id]/class/create

- Takes a POST request and creates a class under a specific studio
- Takes name (TextField), description (TextField), coach (TextField), keywords (TextField), capacity (PositiveIntegerField), enrolled(PositiveIntegerField), startTime (DateTimeField), endTime (DateTimeField), endDate (DateField), recurrence (‘none’, ‘daily’, ‘weekly’)
- If recurrence is daily/weekly, will make instances of classes every 1/7 days at the same time as the first created class until endDate
- Response is for non-recurring class is the class created, and the last class of recurring classes if class is recurring

studios/class/[class id]/edit

- Takes PUT/PATCH request to update the information of a specific class

studios/class/[class id]/cancel/single

- Takes a DELETE request to cancel a single class that hasn’t started yet

studios/class/[class id]/cancel/recurring

- Takes a DELETE request to cancel a all recurring instances of given class class that haven’t started yet