Accounts

Models

CustomUser

- Username: a char field for username, maximum of 15 characters
- First_name: a char field for first name, maximum of 30 characters
- Last_name: a char field for last name, maximum of 30 characters
- Email: a email field for user’s email; it has to be unique
- Avatar: an image field for user’s avatar character; it’s optional
- Password: a char field for user’s password, maximum of 30 characters
- Password2: a char field to confirm user’s password, maximum of 30 characters
- Phone_num: a char field for usr’s phone number, maximum of 15 characters
- Subscription: a foreign key field referencing to user’s subscription plan
- Subscribed: a boolean field indicating if user has subscribed to any plan; defaul is false
- Next_payment_date: a date field for user’s next payment date; default is null

User registration

Anyone who wishes to register can create an account. The username must be greater than 4 characters. The password must be at least 8 characters and the confirmation password must match. The phone number must be a Canadian phone number. If input data don’t meet the requirements, a validation error will be raised. Upon successful registration, a JSON string containing the user’s information will be returned.

- End point: /accounts/register
- Methods: POST
- Payload:
  ```
  {
    "first_name": "string",
    "last_name": "string",
    "username": "string",
    "email": "user@example.com",
    "phone_num": "string",
    "password": "string",
    "password2": "string”
  }
  ```
- response:
  ```
  {
    "first_name": "string",
    "last_name": "string",
    "username": "string",
    "email": "user@example.com",
    "avatar": "http://example.com",
    "phone_num": "string",
    "password": "string",
    "password2": "string"
  }
  ```

User login

Any registered user can log in with their credentials. If the credentials are invalid, a 401 UNAUTHORIZED HTTP response will be returned with the error message "No active account found with the given credentials". Upon successful login, a Bearer token will be returned. For all the views involving authenticated users, this token should be passed.

- End point: /accounts/login
- Methods: POST
- Payload:
  ```
  {
    "username": "string",
    "password": "string"
  }
  ```

Editing profile

Any logged-in user can edit or view their user information. The Bearer token should be passed for users to edit their information. To edit a profile, the phone number must be a Canadian phone number. If input data don't meet the requirements, a validation error will be raised. Upon successful editing, a JSON string containing the user's new information will be returned.

- End point: /accounts/edit/
- Methods: PATCH/GET
- Payload:
  ```
  {
    "email": "user@example.com",
    "first_name": "string",
    "last_name": "string",
    "phone_num": "string"
  }
  ```
- Response:
  ```
  {
    "email": "user@example.com",
    "first_name": "string",
    "last_name": "string",
    "phone_num": "string"
  }
  ```

Resetting password

Any logged-in user can reset his/her password. The Bearer token should be passed for users to reset their password. The old password must match the user's current password. The new password must be at least 8 characters and the confirmation password must match. If input data don't meet the requirements, a validation error will be raised. Upon successful password reset, an empty JSON string will be returned as passwords are confidential.

- End point: /accounts/password/
- Methods: PATCH
- Payload:
  ```
  {
    "old_password": "string",
    "password": "string",
    "password2": "string"
  }
  ```

Studios

Models

Studio

- Name: a char field for studio’s name, maximum of 100 characters
- Address: a char field for studio’s address, maximum of 100 characters
- Latitude: a decimal field for studio’s latitude, maximum of 9 digits and with 6 decimal places
- Longitude: a decimal field for studio’s longitude, maximum of 9 digits and with 6 decimal places
- Postal_code: a char field for studio’s postal code, maximum of 10 characters
- Phone_num: a char field for studio’s phone number, maximum of 20 characters

Amenities

- Studio: a foreign key field referencing the studio which the amenities belong to
- Type: a char field for the type of the amenities, maximum of 50 characters
- Quantity: an integer field for the number of the amenities

Image

- Studio: a foreign key field referencing the studio which the image belongs to
- Image: an image field for studio’s image; it’s optional

List studios by distance

Any user can list studios from closest to furthest. The user must pass valid latitude and longitude. If the geolocation is invalid, a validation error will be raised. A JSON string containing studio names and studio distances from the user will be returned.

- End point: /studios/list
- Method: POST
- Payload:
```
{
  "latitude": "string",
  "longitude": "string"
}
```
- Response:
```
{
    "studio": [
        {
            "name": "string",
            "distance": float
        }
    ]
}
```

List studio details

Any user can list the studio's information. A user_id must be included in the url to indicate which studio to view. A JSON string containing all studio information will be returned, including image urls and amenities.

- End point: /studios/<studio_id>/info
- Method: GET

- Response:
```
[
  {
    "name": "string",
    "address": "string",
    "latitude": "string",
    "longitude": "string",
    "postal_code": "string",
    "phone_num": "string",
    "amenities": [],
    "images": [],
    "classes": []
  }
]
```

Studio search and filter

Any user can search and filter studios based on some conditions. Users can choose one or multiple filters. A JSON string containing all corresponding studio information will be returned.

- End point: /studios/search
- Method: GET
- Params: name, amenities, class_name, coach
- response:
```
[
  {
    "name": "string",
    "address": "string",
    "latitude": "string",
    "longitude": "string",
    "postal_code": "string",
    "phone_num": "string"
  }
]
```

Classes

Recurrence classes of all occurrences

Models

Keyword

- Keyword field, a char field that explains the key things of the class

Class

- Name field, char field to show the name of class
- Description field, text field to show the description of the class
- Coach field, char field to show the name of the coach who teaches the class
- Keywords field, many to many field field to show a list of keywords about the class

- Capacity field, integer field to show how many people can be included in the class
- Schedule day field, integer field which has choices to store the day of the week on which the class will be held
- Start time field, time field to indicate the time when the class begins
- End time field, time field to indicate the time when the class ends
- End recursion date, date filed to indicate the date when the class will end after recurrences

Class Occurrence

- Date field, date field to indicate the date of particular class occurrence
- Students enrolled field, many to many field to indicate a list of students who enrol the class
- Cancelled field, boolean field to indicate if the student has cancelled the class

Enrol in class

The user can enrol the class for one occurrences or all occurrences. The user should choose class id, type of single and the date they want to enrol if they only want to enrol for one occurrence. The users can choose class id and type of all if they want all occurrences.

- End point: /classes/enrol/
- Method: POST
- Payload:
  {“class_id”: “1”,
  “type”: “single”,
  “date: “2022-11-28”}
- response:
  ```
  {
    "detail": "You have successfully enrolled in skip on Nov 28 2022."
  }
  ```

Drop class for one occurrence

The user can drop the class for one occurrences or all occurrences. The user should choose class id, type of single and the date they want to drop if they only want to drop only one occurrence. The users can choose class id and type of all if they want to drop all occurrences.

- End point: /classes/drop/
- Method: POST
- Payload:
  {“class_id”: “1”,
  “type”: “single”,
  “date: “2022-11-28”}
- response:
  ```
  {
    "detail": "You have successfully dropped from skip on Nov 28 2022."
  }
  ```

Class history

Any user can see the class history which should happen in the past. A JSON string containing class name, description of class, coach name, keywords, capacity, occurrence date, start time, and end time will be returned.

- End point: /classes/history/

- Method: GET
- response:
```
{
  "History": [
    {
      "name": "skip",
      "description": "aaa",
      "coach": "dan",
      "keywords": [
        "health"
      ],
      "capacity": "10",
      "date": "2022-11-17",
      "start_time": "05:27 AM",
      "end_time": "06:27 AM"
    }
  ]
}
```

Class schedule

Any user can see the class schedule which should list the classes the user has enrolled. A JSON string containing class name, description of class, coach name, keywords, capacity, occurrence date, start time, and end time will be returned.

- End point: /classes/schedule/
- Method: GET
- response:
```
{
  "Schedule": [
    {
      "name": "skip",
      "description": "aaa",
      "coach": "dan",
      "keywords": [
        "health"
      ],
      "capacity": "10",
      "date": "2022-11-28",
      "start_time": "05:27 AM",
      "end_time": "06:27 AM"
    },
    {
      "name": "skip",
      "description": "aaa",
      "coach": "dan",
      "keywords": [
                "health"
            ],
            "capacity": "10",
            "date": "2022-12-05",
            "start_time": "05:27 AM",
            "end_time": "06:27 AM"
        }
    ]
}
```

Class search and filter

Any logged-in user can search and filter one studio’s classes based on some conditions. Users can choose one or multiple filters. Notice that date follows the format “YYYY-MM-DD”. Start time and end time follow the format “H:M AM/PM”. A JSON string containing all corresponding class schedules will be returned.

- End point: /studios/search
- Method: GET
- Params: studio_name, class_name, coach, date, start_time, end_time
- response:

```
{
  "count": 0,
  "next": "http://example.com",
  "previous": "http://example.com",
  "results": [
    {
      "name": "string",
      "description": "string",
      "coach": "string",
      "keywords": [
        "string"
      ],
      "capacity": "string",
      "date": "2019-08-24",
      "start_time": "string",
      "end_time": "string"
    }
  ]
}
```

Subscriptions

Models

Subscription

- Price field, a char field that holds the price of the subscription per cycle
- Term field, a char field indicating the length of the cycle, it can only be either “MONTH” or “YEAR”

PaymentCard

- Card_holder_name field, a char field that holds the name
- Card_num field, a char field that holds the card number, validated during creation, can only be a 16 digits string
- Expiry_date field, a char field that holds the expiry date of the card, in format MMYY
- Postal_code field, a char field that holds the postal code of the billing address
- Billing_address field, a char field that holds the billing address
- User field, a one to one field referencing a CustomUser, meaning only one card can be created, but the user can update any field after creation

PaymentHistory

- Amount_paid field, a char field that takes only strings that are convertible to a float
- Payment_card field, a foreign key field referencing a payment card
- Datetime field, a datetime field that records the time when the payment was made.
- User field, a foreign key field referencing a CustomUser

Get all subscriptions (login required)

Users can get all possible subscriptions created by the admin, the response returns a price which indicates the price per term, and a term which indicates the cycle length of the subscription, in addition, the term would be either “MONTH” or “YEAR”.

- End point: /subscriptions/all
- Method: GET
- Response:
```
[
  {
    "price": "string",
    "term": "YEAR" or “MONTH”
  }
]
```

Cancel Subscriptions (login required)

Cancel the subscription user subscribed, there will be no future payment and the classes that are booked after the end of the cycle will be removed.

- End point: /subscriptions/cancel-subscription
- Method: POST
- Payload:
- Response:
```
[
  {
    "price": "string",
    "term": "YEAR" or “MONTH”
  }
]
```

Check the Subscription User Subscribed (login required)

Check which subscription the user has subscribed to, price indicates the price per term and term indicates the length of the cycle, term can only be either “YEAR” or “MONTH”.

- End point: /subscriptions/check-subscription
- Method: GET
- Params:

- Response:
```
[
  {
    "price": "string",
    "term": "YEAR" or “MONTH”
  }
]
```

Get all history payments associated with the user (login required)

Get all payment histories associated with the user. Returns the amount paid for each payment, the id of the card used to make the payment, the time of the payment and the user associated with the payment.

- End point: /subscriptions/history
- Method: GET
- Params:
- Response:
```
[
  {
    "amount_paid": "string",
    "payment_card": 0,
    "datetime": "2019-08-24T14:15:22Z",
    "user": 0
  }
]
```

Create a new payment card (login required)

Create a new payment card for the user. Note that only one payment card can be created. However, users can update the information regarding the card. For the payload, the card holder's name must be a string, a 16-digit credit card number, an expiry date in the format of MMYY, the postal code and the billing address.

- End point: /subscriptions/new-payment-card
- Method: POST
- Payload:
```
{
  "card_holder_name": "string",
  "card_num": "string", 16 digits
  "expiry_date": "string", MMYY
  "postal_code": "string",
  "billing_address": "string"
}
```

- Response:
```
{
  "card_holder_name": "string",
  "card_num": "string",
  "expiry_date": "string",
  "postal_code": "string",
  "billing_address": "string",
  "user": 0
}
```

Create a new subscription (admin only)

Create a new subscription for other users to subscribe to. The price must be a string, the term must be either “YEAR” or “MONTH”. Each option will result in a difference in the next payment date.

- End point: /subscriptions/new
- Method: POST
- Payload:
```
{
  "price": "string",
  "term": must be "YEAR" or “MONTH”
}
```
- Response:
```
{
  "price": "string",
  "term": "YEAR"
}
```

Get/update payment card info (login required)

Users can use GET to get the payment card info, PUT or PATCH to update the payment card info. The card holder's name must be a string, a 16-digit card number, an expiry date in the format of MMYY, the postal code and the billing address.

- End point: /subscriptions/payment-card
- Method: GET, PUT, PATCH
- Payload: (PUT, PATCH only)
```
{
  "card_holder_name": "string",
  "card_num": "string", 16 digits
  "expiry_date": "string", MMYY
  "postal_code": "string",
  "billing_address": "string"
}
```
- Response:
```
{
  "card_holder_name": "string",
  "card_num": "string",
  "expiry_date": "string",
  "postal_code": "string",
  "billing_address": "string",
  "user": 0
}
```

Subscribe to the subscription (login required)

User may subscribe to a subscription by providing the subscription number. Users must have a payment card recorded in the system. After subscribing, payment will be immediately initiated and the next payment date will also be updated.

- End point: /subscriptions/subscribe
- Method: POST
- Payload: subscription_id:string
- Response:

"No Subscription Found" or "No Payment Card Found" or "Charge Failed" or "Subscribed Successfully"

Get subscription details by ID (admin only)

The admin can get the subscriptions by id. It will return the price for the subscription and the cycle length “term” for the subscription. “term” must be either “YEAR” or “MONTH”

- End point: /subscriptions/{subscription_id}/
- Method: GET
- Params: subscription_id: string
- Response:
  ```
  {
    "price": "string",
    "term": "YEAR" or “MONTH”
  }
  ```

Delete/update subscription details by ID (admin only)

The admin can update/delete the subscriptions by id. It will return the price for the subscription and the cycle length “term” for the subscription. “term” must be either “YEAR” or “MONTH”

- End point: /subscriptions/{subscription_id}/
- Method: PUT/PATCH/DEL
- Payload:
  ```
  {
    "price": "string",
    "term": “YEAR” or “MONTH”
  }
  ```
- Response:
  ```
  {
    "price": "string",
    "term": “YEAR” or “MONTH”
  }
  ```

Get the next payment date (login required)

Returns the next payment date for the user in a string format, None if not subscribed.

- End point: /subscriptions/next-payment-date/
- Method: GET
- Response:
  "2024-11-21”