CSC309 Project Backend ⚡ 🏋️ 🐻

Team members: Vedant Prajapati, Kyle Blackie

Model Architecture

See Miro board: https://miro.com/app/board/uXjVPGVOeiA=/?share_link_id=14325970418

We have also added photos from the Miro board in the appendix below.

API Endpoints

Users

Register User

Endpoint:

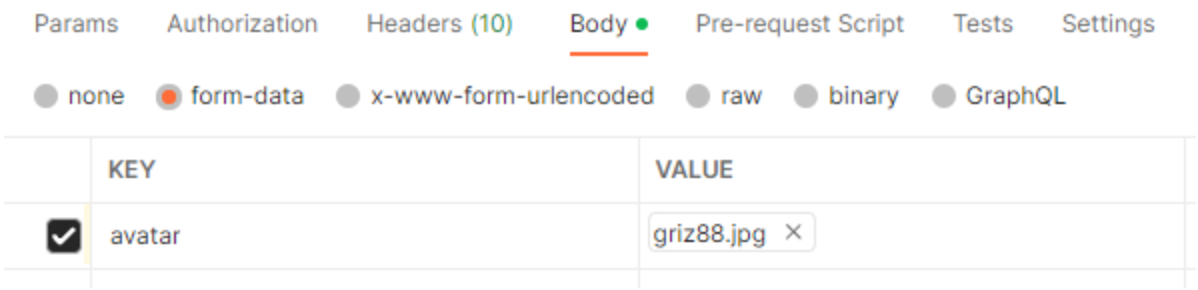
```
POST localhost:8000/users/signup/
```

Payload:

```
{
    "username": String,
    "password": String,
    "first_name": String,
    "last_name": String,
    "email": String,
    "phone_number": String
}
```

Description:

Registers a user using the information provided in the payload
Note: Upload user avatar images using form-data in postman
i.e.,



Edit User Profile

Endpoint:

```
PATCH localhost:8000/users/edit/
```

Note: requires Bearer Token

Payload:

```
{
    "username": String,
    "password": String,
    "first_name": String,
    "last_name": String,
    "email": String,
    "phone_number": String
}
```

Description:

Edit the authenticated user's profile with the payload provided.
Note: Upload user avatar images using form-data in postman

Login User

Endpoint:

```
POST localhost:8000/users/login/
```

Payload:

```
{
    "username": String,
    "password": String
}
```

Description:

Get a JWT access token for a user by providing valid user credentials.

Add Credit/Debit Card Information

Endpoint:

```
POST localhost:8000/users/card/create/
```

Payload:

```
{
    "card_type":"DB",
    "card_number":"126",
    "card_holder_name":"leaf",
    "expiry_date":"2025-01-10T00:00:00Z",
    "security_code":"420"
}
```

Description:

Adds debit/credit card information to a user

Update Credit/Debit Card Information

Endpoint:

```
PUT localhost:8000/users/card/edit/
```

Payload:

```
{
    "card_type":"DB",
    "card_number":"126",
    "card_holder_name":"leaf",
    "expiry_date":"2025-01-10T00:00:00Z",
    "security_code":"420"
}
```

Description:

Updates debit/credit card information of a user

Studios

List Nearby Studios

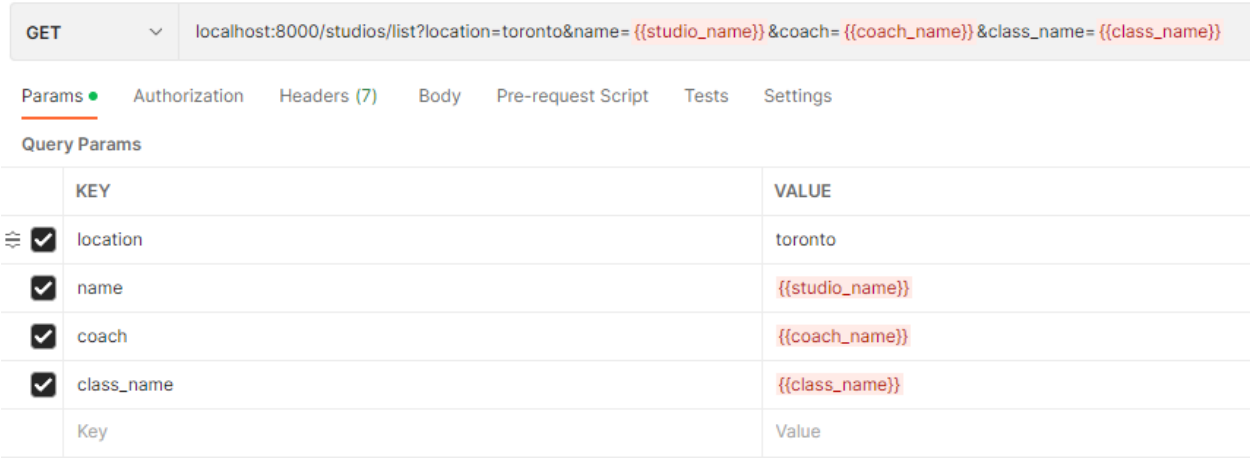
Endpoint:

```
GET localhost:8000/studios/list?location=toronto
```

Payload:

```
{}
```

Note: You can also filter studio results using query parameters



Description:

Returns studios matching filters in order of distance from location query parameter

Individual Studio Info

Endpoint:

```
GET localhost:8000/studios/<int:studio_id>/
```

Payload:

```
{}
```

Description:

Returns information for a single studio.
Note: You can also get directions to the studio by passing in “location” as a query parameter representing your starting location.
I.e.,

```
GET localhost:8000/studios/<int:studio_id>/?location=toronto
```

Subscriptions

Get all the available subscription options

Endpoint:

```
GET localhost:8000/subscriptions/list/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Display all the subscription models that users can subscribe to.

Subscribe to subscription

Endpoint:

```
PATCH localhost:8000/subscriptions/subscribe/<int: subscription_id>/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Subscribe a user to a subscription and make their initial payment

Cancel Current Subscription

Endpoint:

```
PATCH localhost:8000/subscriptions/cancel/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Cancel the authenticated user’s subscription if it exists

Get Payment History

Endpoint:

```
GET localhost:8000/subscriptions/payments/history/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Returns the user's previously made payments.

Get Upcoming Payments

Endpoint:

```
GET localhost:8000/subscriptions/payments/future/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Returns info about the user's upcoming payments.

Python Function to Make all user payments that are due today

Function can be found in subscriptions/views. Please run:

```
cd TorontoFitnessClub
./manage.py shell
>> from subscriptions.views import pay_user_subscriptions_due_today
```

```
>> pay_user_subscriptions_due_today
```

Function Code:

```
def pay_user_subscriptions_due_today():
      users = User.objects.all()
      if not users:
        print("No users found")
        return

      for user in users:
      if user.subscription:
            payment = user.payments.order_by("-date").first()
            if (
                  payment
                  and payment.date.date()
                  + relativedelta(months=user.subscription.term_length)
                  == datetime.date.today()
                  and user.card
            ):
                  # pay the subscription
                  Payment.objects.create(
                  user=user,
                  amount=user.subscription.cost,
                  card_holder_name=user.card.card_holder_name,
                  card_type=user.card.card_type,
                  card_number=user.card.card_number,
                  expiry_date=user.card.expiry_date,
                  security_code=user.card.security_code,
                  )
                  print(f"Paid {user.subscription.cost} for
{user.username}")
            else:
                  print(f"Card invalid for {user.username} or payment not
due")
      else:
            print(f"No subscription for {user.username}")
```

Classes

Display the classes taught in a specified studio

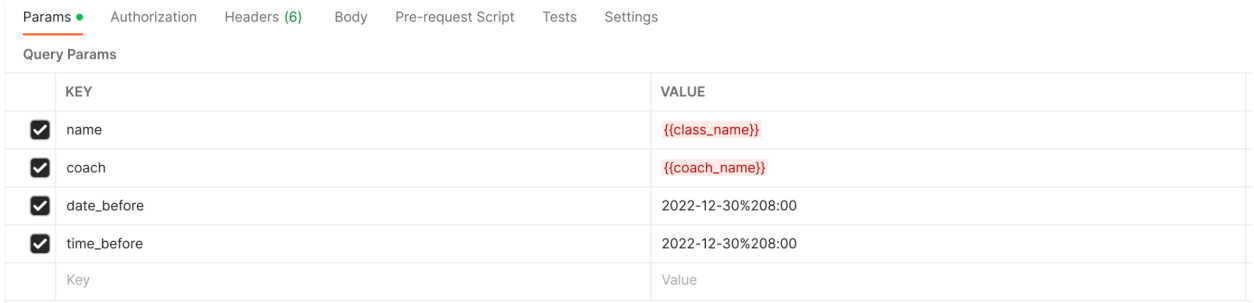
Endpoint:

```
GET localhost:8000/classes/<int: studio_id>/schedule/
```

Payload:

```
{}
```

Query Params for filtering:



Description:

Display all the class sessions matching the user's query parameters for a given studio

Display the upcoming class schedule for a user

Endpoint:

```
GET localhost:8000/classes/schedule/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Display all the upcoming class sessions for the user that are not cancelled

Display the user’s class history

Endpoint:

```
GET localhost:8000/classes/history/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Display all the past class sessions that a user was in enrolled in

Enrol in a class

Endpoint:

```
PATCH localhost:8000/classes/class/<int:class_id>/enrol/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Enrols a user in all upcoming class sessions whose class matches the specified class id

Drop a class

Endpoint:

```
PATCH localhost:8000/classes/class/<class_id>/drop/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Drops all upcoming class sessions of a class from a user

Enrol in a class session

Endpoint:

```
PATCH localhost:8000/classes/session/<int: class_session_id>/enrol/
```

Note: requires Bearer Token

Payload:

```
{}
```

Description:

Enrols a user in an upcoming class session whose class

Drop a class session

Endpoint:

```
PATCH localhost:8000/classes/session/<int: class_session_id>/drop/
```

Note: requires Bearer Token

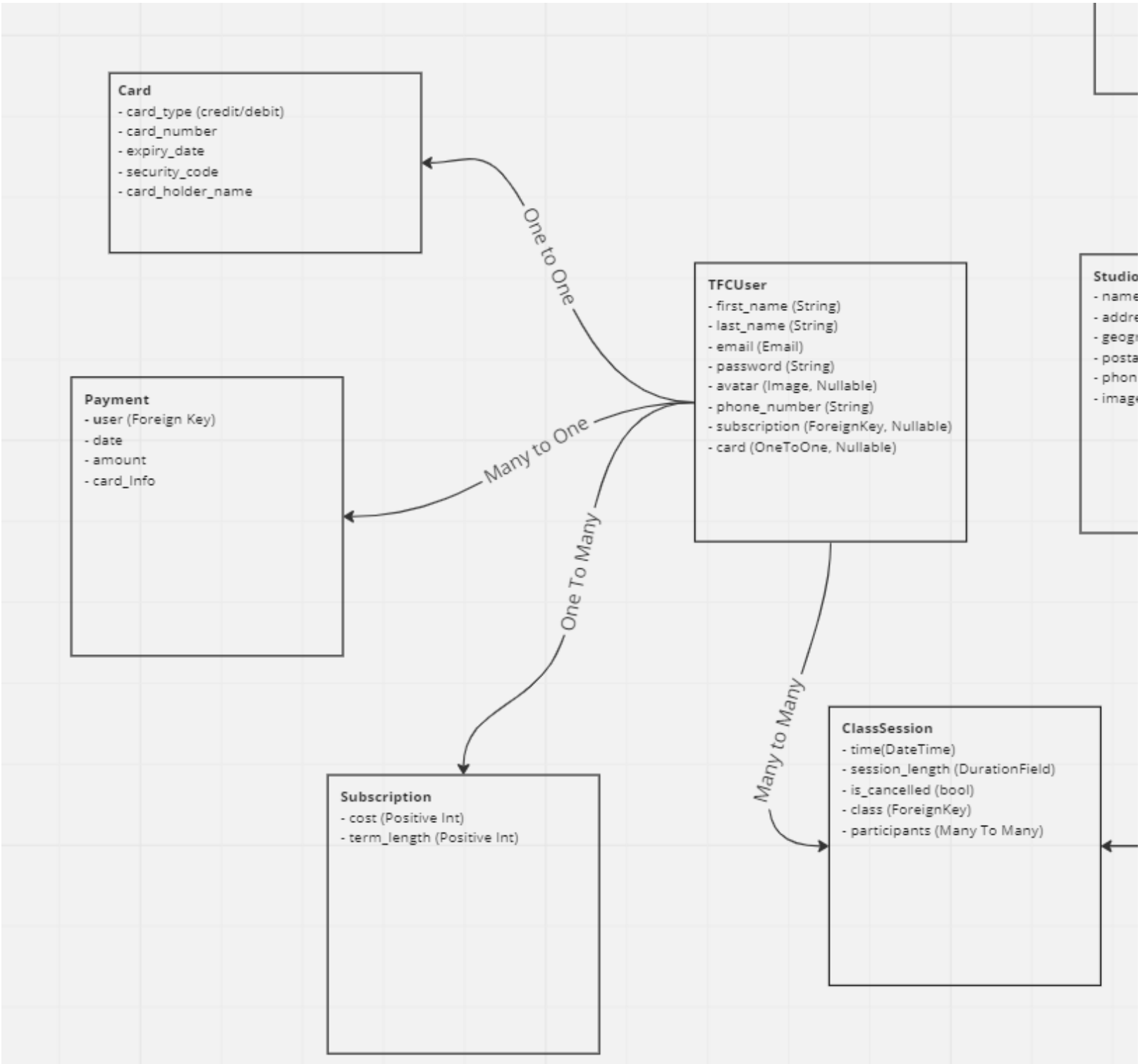
Payload:

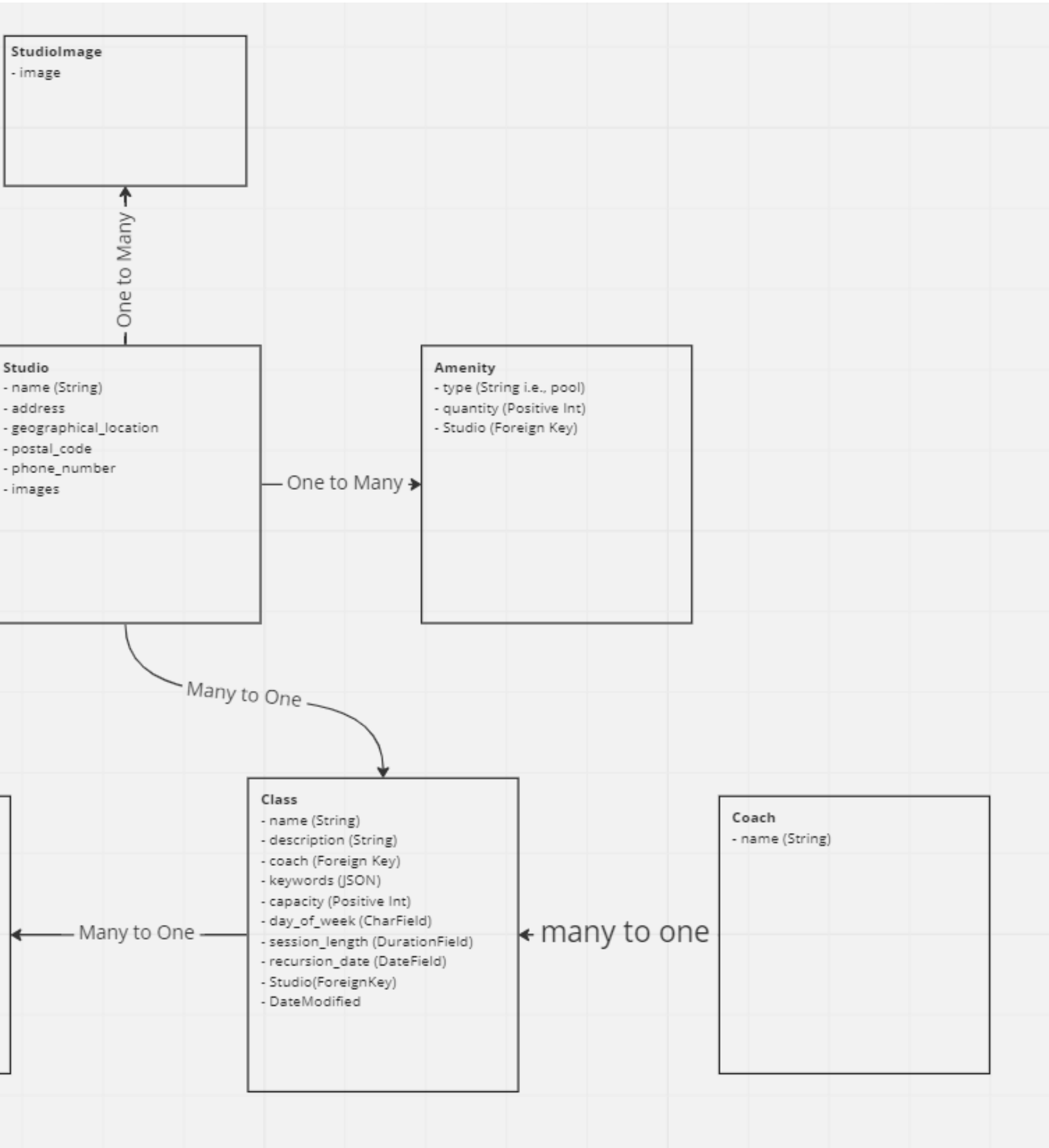
```
{}
```

Description:

Drop a user from a class session matching the session_id

Appendix:


Card
- card_type (credit/debit)
- card_number
- expiry_date
- security_code
- card_holder_name
One to One
TFCUser
- first_name (String)
- last_name (String)
- email (Email)
- password (String)
- avatar (Image, Nullable)
- phone_number (String)
- subscription (ForeignKey, Nullable)
- card (OneToOne, Nullable)
Payment
- user (Foreign Key)
- date
- amount
- card_Info
Many to One
One To Many
Many to Many
ClassSession
- time(DateTime)
- session_length (DurationField)
- is_cancelled (bool)
- class (ForeignKey)
- participants (Many To Many)
Subscription
- cost (Positive Int)
- term_length (Positive Int)


StudioImage
- image
One to Many
Studio
- name (String)
- address
- geographical_location
- postal_code
- phone_number
- images
One to Many
Amenity
- type (String i.e., pool)
- quantity (Positive Int)
- Studio (Foreign Key)
Many to One
Class
- name (String)
- description (String)
- coach (Foreign Key)
- keywords (JSON)
- capacity (Positive Int)
- day_of_week (CharField)
- session_length (DurationField)
- recursion_date (DateField)
- Studio(ForeignKey)
- DateModified
Many to One
many to one
Coach
- name (String)