PB DOCUMENTATION

Models

Profile (extend base User):

Fields (not including the ones already in User):

1. Avatar (ImageField)
2. Phone (CharField)
3. Current_subscription (OneToOneField) [one-to-one with Subscription]
4. Card_info (CharField)

Plan:

Fields:

1. Name (CharField)
2. Price (DecimalField)
3. Billing_period (CharField)

Subscription:

Fields:

1. User (ForeignKey) [one-to-one with Profile]
2. Plan (ForeignKey) [one-to-one witH Studio]
3. Date_of_purchase (DateTimeField)
4. Next_payment (DateTimeField)
5. Card_info (CharField)

Studio:

Fields:

1. Name (CharField)
2. Address (CharField)
3. Latitude (DecimalField)
4. Longitude (DecimalField)
5. Postal_code (CharField)
6. Phone_num (CharField)

Amenities

Fields:

1. Name (CharField)
2. Quantity (PositiveIntegerField)
3. Studio (ForeignKey) [many-to-one with Studio]

StudioImages

Fields:

1. Img (ImageField)
2. Studio [many-to-one] (ForeignKey)

Klass (named liked so to avoid conflict with python's class):

Fields:

1. Name (CharField)
2. Description (TextField)
3. Coach (CharField)
4. Keywords = (TextField)
5. Capacity (PositiveIntegerField)
6. Studio = (ForeignKey) [many-to-one with Studio]
7. Day (CharField)
8. Start_time (TimeField)
9. End_time (TimeField)
10. First_day (DateField)
11. End_recursion (DateField)
12. <u>Date (DateField)</u>
13. <u>User (ManyToManyField) [many-to-many with Profile]</u>
14. <u>Next_klass (ForeignKey) [many-to-one with Klass]</u>

Note: Underlines Fields have editable=False so that the admin does not have to provide anything for them. User is modified after enroll/dropout requests; for more information on the other underlined fields, see KlassAdmin

KlassAdmin

Note: Overrides Admin's save_model by creating all necessary instances of Klass. Sets Klass' date based on Klass' day and Klass' first_day.. Specifically, when creating a Klass (say, klassA) with day Monday, first_day being next week's Monday, end_recursion 4 weeks from now, it will create two more, looking like this

klassA: (date=next week's Monday, next_klass=klassB)
klassB: (date=next next week's Monday, next_klass=klassC)
klassC: (date=next next next week's Monday, next_klass=None)

APIs **(Note: All required query parameters are in bold)**

Studios

View Studio:

Method: GET

Endpoint: /studios/<studio_id>/

Query Parameters: None

Fields/Payload:

1. name
2. address
3. longitude
4. latitude
5. postal_code
6. phone_num
7. images
8. klass
9. amenities

View Studio's Classes:

Method: GET

Endpoint: /studios/<studio_id>/classes

Query Parameters (all the following are regarding a class, not a studio):

1. name
2. day (Monday, Tuesday, etc., NOT e.g. 2022-11-15)
3. coach
4. start_time (in the form of 23:59)
5. end_time (in the form of 23:59)

Fields/Payload:

1. name
2. description
3. coach
4. capacity
5. keywords
6. studio
7. day
8. start_time

9. end_time
10. end_recursion
11. user
12. date

View All Studios:

Method: GET

Endpoint: /studios/all/

Query Parameters (all the following are regarding the studio):

1. name
2. class_name
3. coach
4. amenities (only type amenity's name, no quantity)
5. **latitude**
6. **longitude**

Fields/Payload:

1. name
2. address
3. longitude
4. latitude
5. postal_code
6. phone_num
7. images
8. klass
9. amenities

Classes

Enroll One Instance of a Class:

Method: PUT

Endpoint: /classes/<class_id>/enroll/

Query Parameters: None

Fields/Payload: None

Note: User must have subscription, not already enrolled, and class must have a slot for an extra user to join.

Dropout One Instance of a Class:

Method: PUT

Endpoint: /classes/<class_id>/dropout/

Query Parameters: None

Fields/Payload: None

Note: User must be already enrolled

Enroll One Instance of a Class:

Method: PUT

Endpoint: /classes/<class_id>/enroll/all/

Query Parameters: None

Fields/Payload: None

Note: User must have subscription, not already enrolled, and class must have a slot for an extra user to join for **all** classes

Dropout One Instance of a Class:

Method: PUT

Endpoint: /classes/<class_id>/dropout/all/

Query Parameters: None

Fields/Payload: None

Note: User must be already enrolled in **all** classes

Profiles

Register Profile:

Method: POST

Endpoint: /accounts/signup/

Query Parameters: None

Fields/Payload:

1. username
2. first_name
3. last_name
4. email
5. password
6. password2
7. phone
8. avatar

N0te: Only username, password and password2 are required

Login:

Method: POST

Endpoint: /accounts/login

Query Parameters: None

Fields/Payload:

1. username
2. password

Edit Profile:

Method: PATCH

Endpoint: /accounts/edit/

Query Parameters: None

Fields/Payload: None

1. username
2. first_name
3. last_name
4. email

5. password
6. phone
7. avatar

Make subscription:

Method: POST

Endpoint: /accounts/subscribe/

Query Parameters: None

Fields/Payload:

1. plan
2. card_info

Note:

- plan is the name of the plan
- card_info is not necessary if user already has a card in its profile, if card_info is provided in payload, the new card_info will be used instead.

Update Card:

Method: POST

Endpoint: /accounts/card_update/

Query Parameters: None

Fields/Payload: card_info

Update Subscription:

Method: POST

Endpoint: /accounts/subscription_update/

Query Parameters: None

Fields/Payload: plan, card_info

Cancel Subscription:

Method: POST

Endpoint: /accounts/subscription_cancel/

Query Parameters: None

Fields/Payload: None

View Payment History:

Method: GET

Endpoint: /accounts/payment_history/

Query Parameters: None

Fields/Payload: None