CSC309 Phase 2 REST API Documentation

Authentification:

Models:

```
class User(AbstractUser):
   account_type = models.TextField(choices=(('seeker', 'Seeker'), ('shelter',
'Shelter')))
   avatar = models.ImageField(upload_to='media/')

class Seeker(models.Model):
   user = models.OneToOneField(User, primary_key=True,
on_delete=models.CASCADE)

   def delete(self, *args, **kwargs):
       self.user.delete()
       return super().delete(*args, **kwargs)

class Shelter(models.Model):
   user = models.OneToOneField(User, primary_key=True, on_delete =
models.CASCADE)
   name = models.TextField(max_length=200)
   address = models.TextField(max_length=200)

   def delete(self, *args, **kwargs):
       self.user.delete()
       return super().delete(*args, **kwargs)
```

`/token`

- Payload:
  - Username
  - Password
- POST: sign in as a shelter or a seeker and get a token.

`/seeker`

- Payload:
  - Username
  - Password1
  - Password2
  - Email
  - Name
  - Avatar

- `POST`: create a new seeker
    - With the entire payload, create a new seeker. Does not need to be authenticated.
- `PATCH`: update a seeker's information
    - Only the seeker should be able to update their own information. Updates the seekers information and requires authentication.
    - Password1 and Password2 are not mandatory, when they are omitted then the password stays the same.
    - Avatar is not mandatory, when it is missing the avatar stays the same.
- `DELETE`: delete a seeker
    - Only the seeker should be able to delete itself.
- `GET`: get the logged in seeker information
    - Requires authentication

`/seeker/<int:pk>`

- `GET`: Get the seeker pk information
    - Requires authentication as a shelter. Needs an application from the seeker for one of the shelter's pets.

`/shelter`

- Payload:
    - Username
    - Password1
    - Password2
    - Name
    - Address
    - Email
    - avatar
- `POST`: create a new shelter
    - With the entire payload, create a new shelter. Does not need to be authenticated.
- `PATCH`: update a shelter's information
    - Only the shelter should be able to update their own information. Updates the shelter's information and requires authentication.
    - Password1 and Password2 are not mandatory, when they are omitted then the password stays the same.
    - Avatar is not mandatory, when it is missing the avatar stays the same.
- `DELETE`: delete a shelter
    - Only the shelter should be allowed to delete itself.

- `GET`: get the logged in seeker information
    - Requires authentication.

`/shelter/<int:pk>`

- `GET`: get the shelter with id *<pk>* information.
    - No authentication required.

`/shelters`

- `GET`: get a list of shelters.
    - No authentication required.

Pet Listings App:

Model:

```
class Pet(models.Model):
   name=models.CharField(max_length=120)
   age=models.IntegerField()
   breed=models.CharField(max_length=120)
   species=models.CharField(max_length=120)
   gender=models.CharField(max_length=120)
   image=models.ImageField(upload_to="uploads", null=True)
   status=models.CharField(max_length=120, null=True)
   size=models.IntegerField(default=0)
   days_on_petpal=models.IntegerField(default=0)
   color=models.CharField(max_length=120, null=True)
   shelter=models.ForeignKey(Shelter, on_delete=models.SET_NULL, null=True)
```

Endpoints:

/petlistings/pets

- Endpoint for viewing multiple pets and creating them
- Method: `PetsListCreate`
- Payload (only for create):
    - name
    - age
    - breed
    - species
    - gender
    - image
    - status -> valid values: ‘available’, ‘adopted’, ‘pending’ and ‘withdrawn’
    - size
    - days_on_petpal,
    - color.

    - shelter -> Only shelters can create pets
- `GET`: get all pet listings
    - Accepts query parameters for sorting and filtering on `GET` request
        - `?ordering=<string>` - can order by 'name' or 'age' (name is default)
        - `?shelter=<id>` - Can pass in id of shelter to get pets that belong to a specific shelter
        - ?status=<string> - Filter by status, options are `'available'`, `'pending'`, `'adopted'`, `'withdrawn'`. **Default status filter is** `'available`, but can pass `status=all` to get all pets, regardless of status
        - ?age=<int>
        - ?species=<string>
    - Accepts query parameters for pagination as well
        - ?page_size=<int> - Specify number of results per page
        - ?p=<int> Specify page number

- `POST`: Creates a new listing using the request data, returns JSON object
    - **Only a shelter is allowed to create pets**
    - To create a Pet in Postman you should specify the request body using form-data to allow uploading image files
    - We have a postman pre request script you can use to help generate pets with random values (feel free to modify this)
        https://gist.github.com/is-ahmed/5f3b254c119a6a7fbe514271773c01ac
        - You can use the variables in the form-data request body with {{randomAge}}, {{randomGender}} etc. for the values

/petlistings/pets/<int:pk>

- Endpoint for retrieving, updating and/or deleting a specific pet
- Fields: pk - id of pet
- Methods: `PetRetrieveUpdateDestroy`
- Payload:
    - All fields for Pet (only for `PATCH`)
- `DELETE`: Delete the pet listing with the given id
    - A shelter can only delete their own pets
    - Seekers can't delete pet of shelter
- `PATCH`: Update a pet listing using the request data
    - A shelter can only update their own pets

- GET: Get data for a pet with that id
    - Any authenticated user can request pet information

Application App:

Model:

```
class Application(models.Model):
   applicationStatus = [
       ('pending', 'Pending'),
       ('accepted', 'Accepted'),
       ('denied', 'Denied'),
       ('withdrawn', 'Withdrawn'),
   ]
   pet_listing = models.ForeignKey(Pet, on_delete=models.CASCADE,
related_name='pet_applications')
   user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,
related_name='user_applications') #AdopterUser
   shelter = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='shelter_applications') #ShelterUser
   status = models.CharField(max_length=10, choices=applicationStatus,
default='pending')
   creation_time = models.DateTimeField(auto_now_add=True)
   last_update_time = models.DateTimeField(auto_now=True)
```

Endpoints:

/applications/

- Requires authentication
- Returns full list of applications for the seeker/shelter currently logged in
- Method: ApplicationListView
- Payload: None
- GET
    - Seeker: Gets all adoption applications
    - Shelter: Gets all adoption applications for any of it’s pets
    - Accepts query for filtering by status
        - ?status=<string>
        - Acceptable values: 'pending', 'accepted', 'denied', 'withdrawn'
    - Accepts query for sorting
        - Orders by creation_time by default
        - ?ordering=<string>
        - Acceptable values: ‘creation_time’, ‘last_update_time’
    - Accepts query for pagination
        - Allows for custom size of how many applications can appear on one page

        - page_size specifies the number of applications per page, and page specifies the page
        - ?page_size=<int>&page=<int>

/applications/<int:pk>/

- Requires authentication
- Returns or updates the application with the id of pk
- Fields: pk - id of the application
- Method: ApplicationRetrieveUpdateView
- Payload (only for PATCH):
    - status:
        - Acceptable values: 'pending', 'accepted', 'denied', 'withdrawn'
        - Also depends on if seeker or shelter is logged on
- PATCH: Updates status of application with id of pk to the value specified in the payload if possible.
- GET: Gets application with id of pk

/petlistings/pets/<int:pet_id>/applications/

- Requires authentication
- Only a seeker is allowed to create applications
- Creates application for pet with id of pet_id if that pet exists and is available and the seeker logged in does not already have an application for that pet
- Method: ApplicationCreateAPIView
- Fields: pet_id - id of the pet the application wants to be created for
- Payload: None
- POST: Create an application for the pet with the id of pet_id if it exists and is available. This is the application for the currently logged-in seeker.

-

Comments App: (Eric)

URL:

```
from django.urls import path
```

```
from .views import UserCommentCreate, ShelterCommentsListView,
ApplicationCommentsListView, ShelterCommentsSortedListView,
ApplicationCommentsSortedListView

urlpatterns = [
   path('commentcreation/<str:content_type>/<int:object_id>/',
UserCommentCreate.as_view(), name='create_comment'),
   path('shelters/<int:shelter_id>/comments/', ShelterCommentsListView.as_view(),
name='shelter_comments'),
   path('shelters/<int:shelter_id>/comments/sorted/',
ShelterCommentsSortedListView.as_view(), name='shelter_comments_sorted'),
   path('applications/<int:application_id>/comments/',
ApplicationCommentsListView.as_view(), name='application_comments'),
   path('applications/<int:application_id>/comments/sorted/',
ApplicationCommentsSortedListView.as_view(), name='application_comments_sorted'),

]
```

Comment model:

```
class Comment(models.Model):
   author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='comments')
   text = models.CharField(max_length=200)
   date_created = models.DateTimeField(default=timezone.now)

   content_type = models.ForeignKey(ContentType, on_delete=models.CASCADE)
   object_id = models.PositiveIntegerField()
   content_object = GenericForeignKey('content_type', 'object_id')
```

Endpoints:

commentcreation/<str:content_type>/<int:object_id>/

- if user or shelter wants to comment on application content_type part must be “application”. Otherwise, this part should be “shelter” with corresponding object id
- Requires token authentication
- Method: UserCommentCreate(CreateAPIView) (POST method)
- Payload: “text”

shelters/<int:shelter_id>/comments/ applications/<int:application_id>/comments/

- Anyone can see particular shelters’ comments with the shelter id key.
- Only users who made requests or shelter requests can see application comments.

- Method: ShelterCommentsListView(ListCreateAPIView), ApplicationCommentsListView(ListCreateAPIView) GET method.
- Payload: None

shelters/<int:shelter_id>/comments/sorted/
applications/<int:application_id>/comments/sorted

- Creates sorted lists mentioned above
- Method: ApplicationCommentsSortedListView(ApplicationCommentsListView), ShelterCommentsSortedListView(ShelterCommentsListView): GET method.
- Payload: None

Notifications App:

Model :

```
TYPE_CHOICES = (
    ("CONVERSATION", "conversation"),
    ("APPLICATION_STATUS", "application status update"),
    ("COMMENT", "new comments"),
    ("REVIEWS", "new reviews"),
    ("NEW_PET_LISTING", "new pet added")
)

# Create your models here.
class Notification(models.Model):
    ...
    type=models.TextField(choices=TYPE_CHOICES)
    read=models.BooleanField(default=False)
    creation_time=models.DateTimeField(auto_now_add=True)
    for_user=models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE);
    link=models.CharField(max_length=200)
```

/notifications/notifs

- Returns notifications for currently logged in user
- Method: GET
- Payload: None
- GET : Get all notifications for the current user
    - Accepts the following query parameters for filtering and ordering
        - `?ordering=<string>`
- No creation endpoint, notifications are created in other parts of the code i.e. when a new pet listing is created, new comment on application etc

/notifications/notifs/<int:pk>

- Returns or deletes the notification with id pk
- Fields: pk - id of notification
- Methods: `GET, DELETE, PATCH`
- `PATCH`: Update notification status from unread to read
- `DELETE`: Only allow users to delete their own notification
- `GET`: Get the notification, returns link to page notification leads to as JSON response