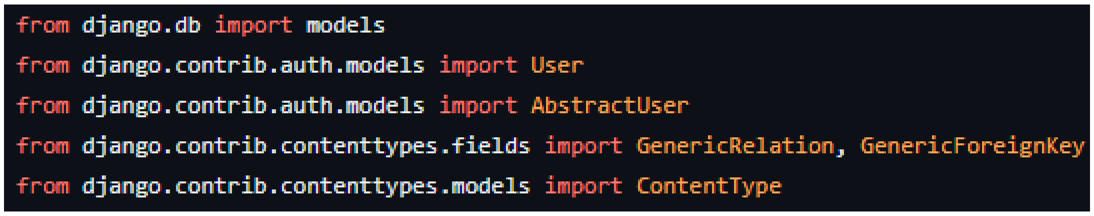
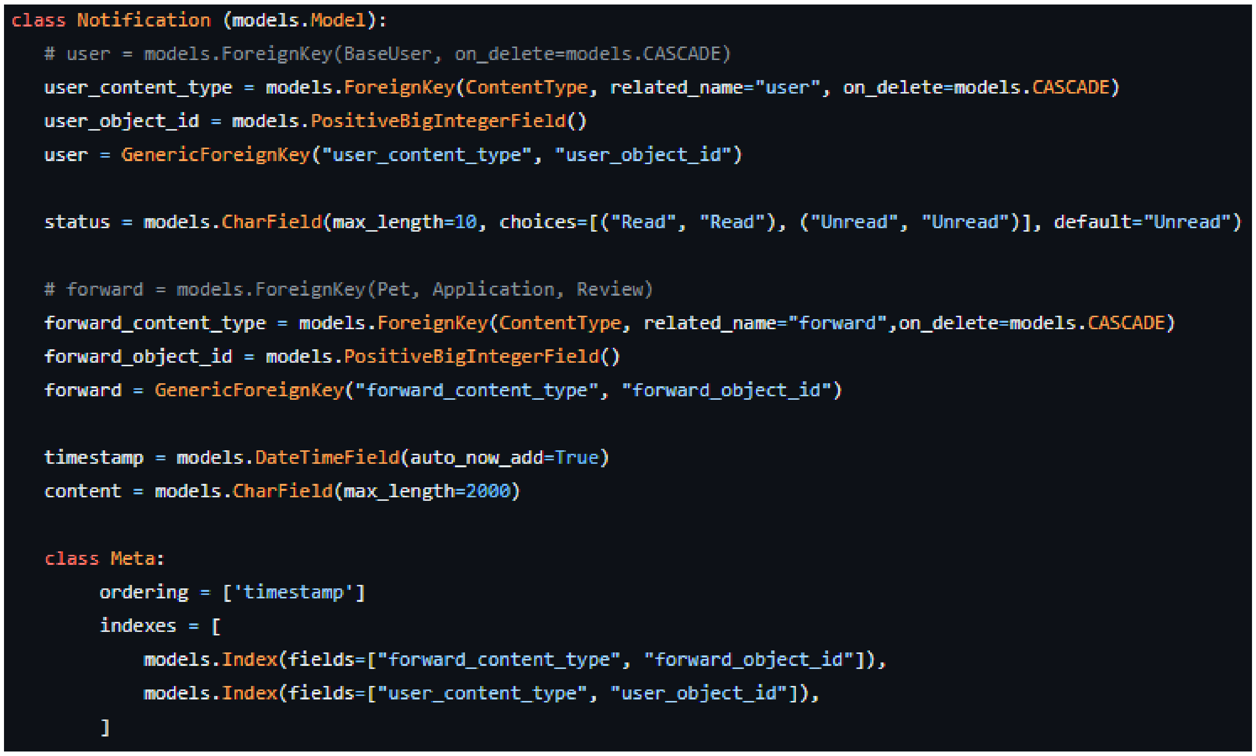
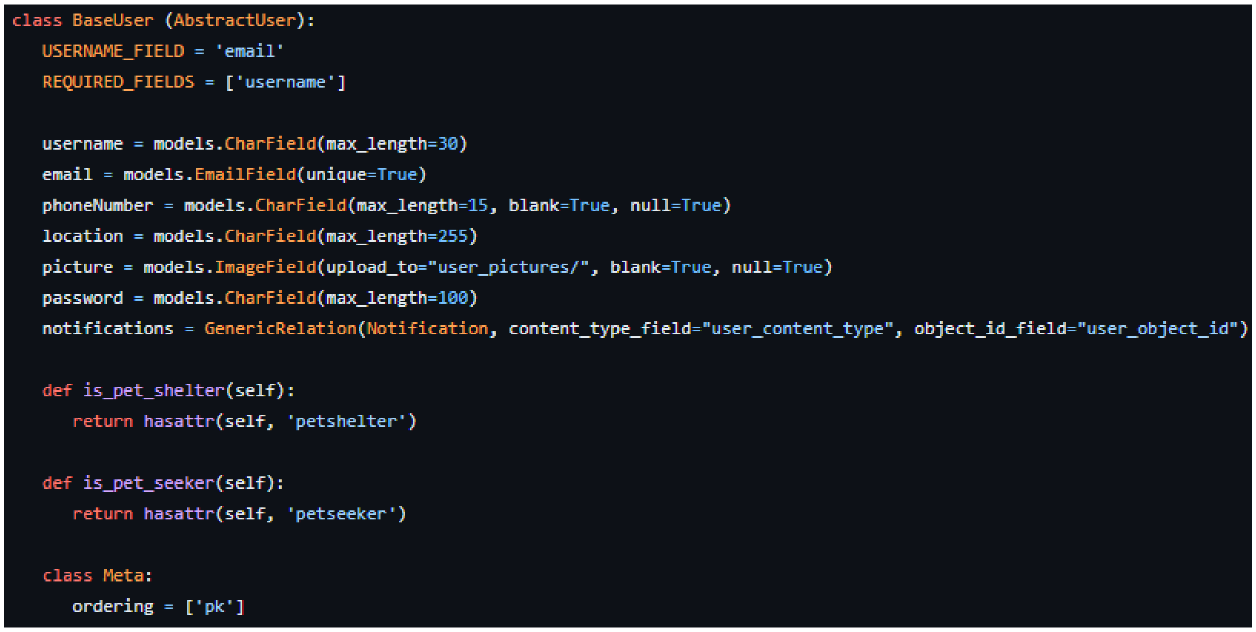
Model:



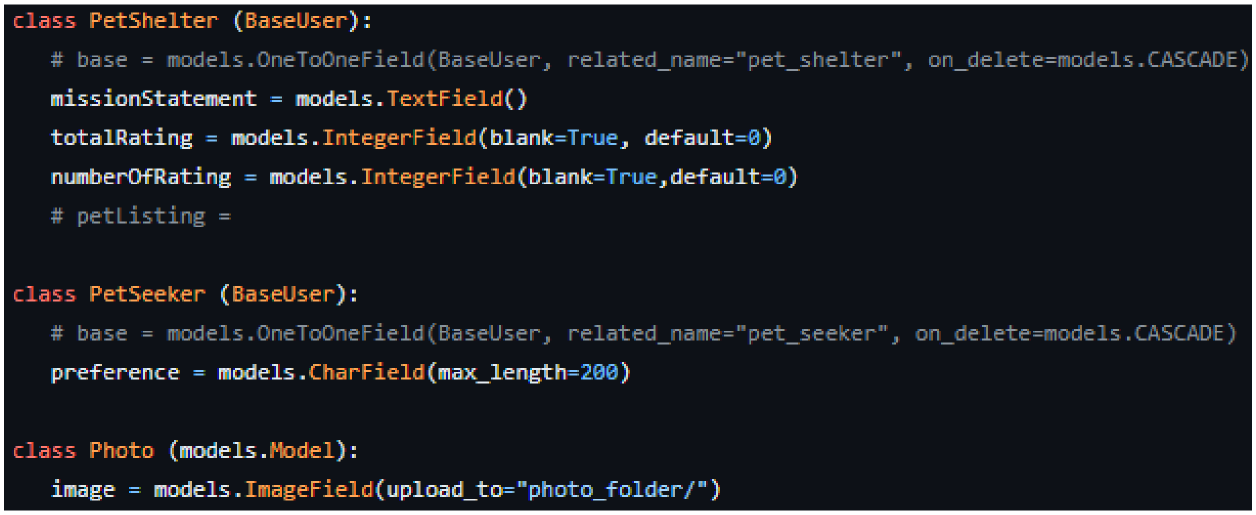
```python
from django.db import models
from django.contrib.auth.models import User
from django.contrib.auth.models import AbstractUser
from django.contrib.contenttypes.fields import GenericRelation, GenericForeignKey
from django.contrib.contenttypes.models import ContentType
```



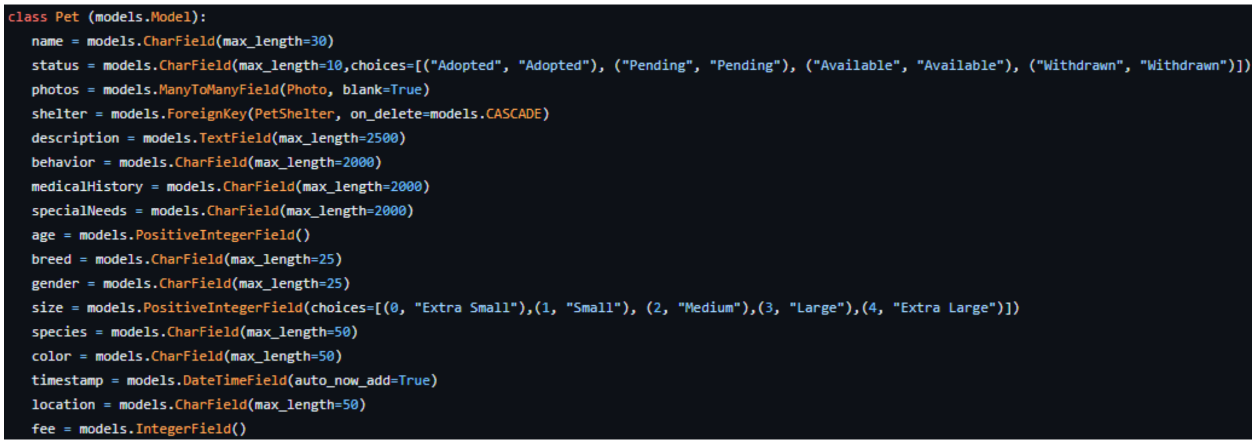
```python
class Notification (models.Model):
    # user = models.ForeignKey(BaseUser, on_delete=models.CASCADE)
    user_content_type = models.ForeignKey(ContentType, related_name="user", on_delete=models.CASCADE)
    user_object_id = models.PositiveBigIntegerField()
    user = GenericForeignKey("user_content_type", "user_object_id")

    status = models.CharField(max_length=10, choices=[("Read", "Read"), ("Unread", "Unread")], default="Unread")

    # forward = models.ForeignKey(Pet, Application, Review)
    forward_content_type = models.ForeignKey(ContentType, related_name="forward",on_delete=models.CASCADE)
    forward_object_id = models.PositiveBigIntegerField()
    forward = GenericForeignKey("forward_content_type", "forward_object_id")

    timestamp = models.DateTimeField(auto_now_add=True)
    content = models.CharField(max_length=2000)

    class Meta:
        ordering = ['timestamp']
        indexes = [
            models.Index(fields=["forward_content_type", "forward_object_id"]),
            models.Index(fields=["user_content_type", "user_object_id"]),
        ]
```



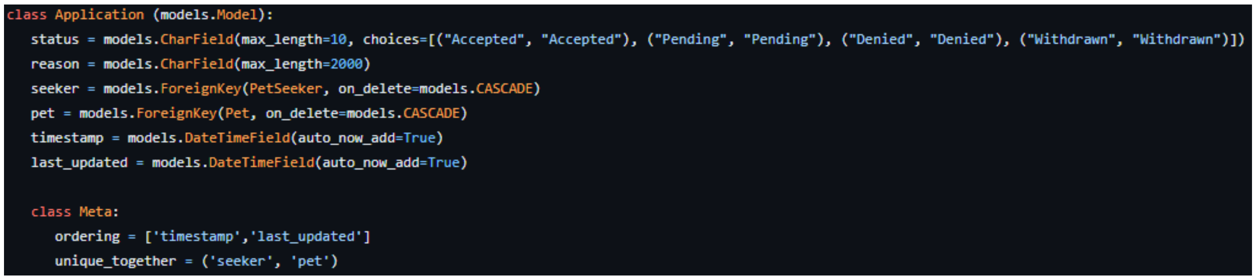
```python
class BaseUser (AbstractUser):
    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['username']

    username = models.CharField(max_length=30)
    email = models.EmailField(unique=True)
    phoneNumber = models.CharField(max_length=15, blank=True, null=True)
    location = models.CharField(max_length=255)
    picture = models.ImageField(upload_to="user_pictures/", blank=True, null=True)
    password = models.CharField(max_length=100)
    notifications = GenericRelation(Notification, content_type_field="user_content_type", object_id_field="user_object_id")

    def is_pet_shelter(self):
        return hasattr(self, 'petshelter')

    def is_pet_seeker(self):
        return hasattr(self, 'petseeker')

    class Meta:
        ordering = ['pk']
```



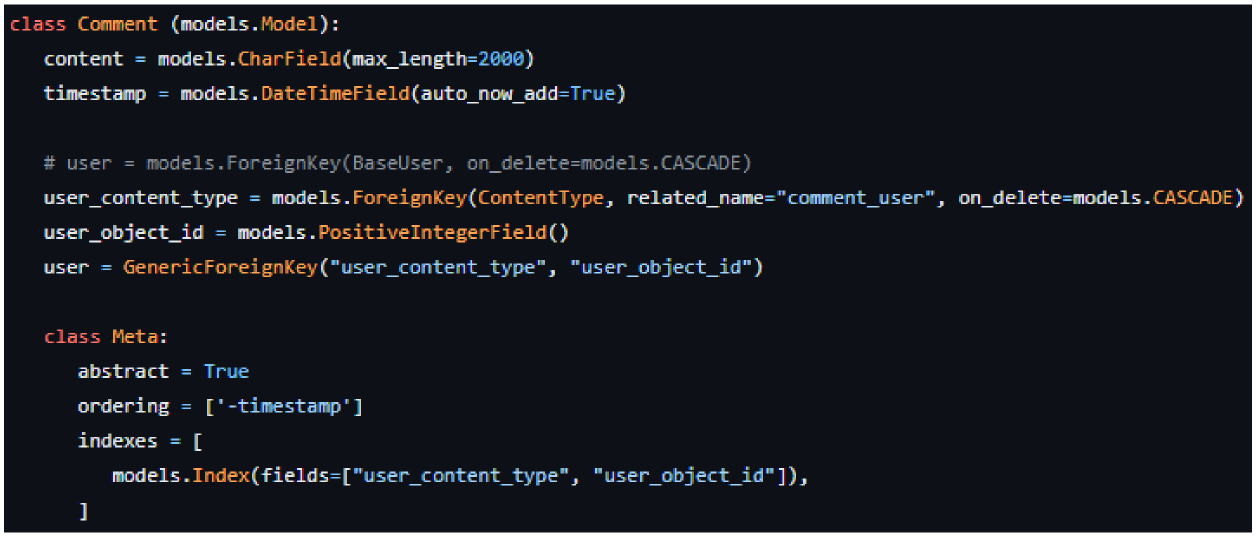
```python
class PetShelter (BaseUser):
    # base = models.OneToOneField(BaseUser, related_name="pet_shelter", on_delete=models.CASCADE)
    missionStatement = models.TextField()
    totalRating = models.IntegerField(blank=True, default=0)
    numberOfRating = models.IntegerField(blank=True,default=0)
    # petListing =

class PetSeeker (BaseUser):
    # base = models.OneToOneField(BaseUser, related_name="pet_seeker", on_delete=models.CASCADE)
    preference = models.CharField(max_length=200)

class Photo (models.Model):
    image = models.ImageField(upload_to="photo_folder/")
```



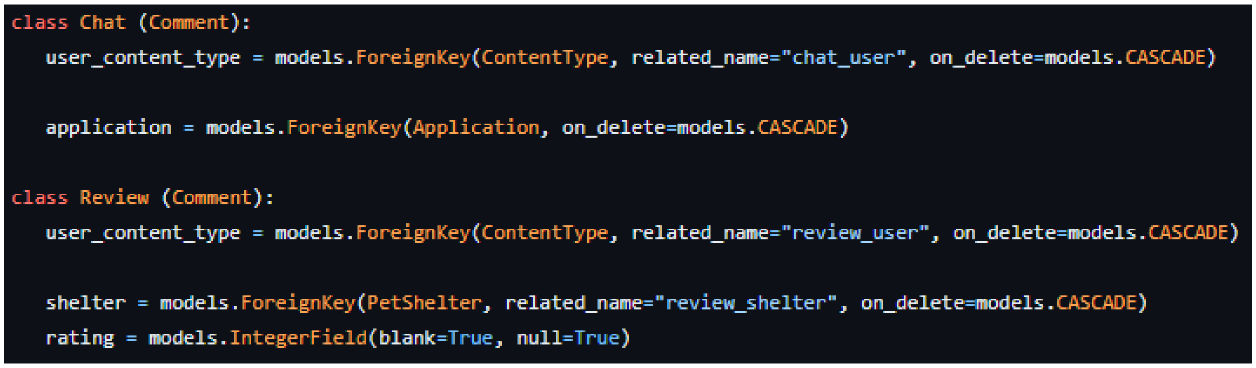
```python
class Pet (models.Model):
    name = models.CharField(max_length=30)
    status = models.CharField(max_length=10,choices=[("Adopted", "Adopted"), ("Pending", "Pending"), ("Available", "Available"), ("Withdrawn", "Withdrawn")])
    photos = models.ManyToManyField(Photo, blank=True)
    shelter = models.ForeignKey(PetShelter, on_delete=models.CASCADE)
    description = models.TextField(max_length=2500)
    behavior = models.CharField(max_length=2000)
    medicalHistory = models.CharField(max_length=2000)
    specialNeeds = models.CharField(max_length=2000)
    age = models.PositiveIntegerField()
    breed = models.CharField(max_length=25)
    gender = models.CharField(max_length=25)
    size = models.PositiveIntegerField(choices=[(0, "Extra Small"),(1, "Small"), (2, "Medium"),(3, "Large"),(4, "Extra Large")])
    species = models.CharField(max_length=50)
    color = models.CharField(max_length=50)
    timestamp = models.DateTimeField(auto_now_add=True)
    location = models.CharField(max_length=50)
    fee = models.IntegerField()
```



```python
class Application (models.Model):
    status = models.CharField(max_length=10, choices=[("Accepted", "Accepted"), ("Pending", "Pending"), ("Denied", "Denied"), ("Withdrawn", "Withdrawn")])
    reason = models.CharField(max_length=2000)
    seeker = models.ForeignKey(PetSeeker, on_delete=models.CASCADE)
    pet = models.ForeignKey(Pet, on_delete=models.CASCADE)
    timestamp = models.DateTimeField(auto_now_add=True)
    last_updated = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['timestamp','last_updated']
        unique_together = ('seeker', 'pet')
```



```python
class Comment (models.Model):
   content = models.CharField(max_length=2000)
   timestamp = models.DateTimeField(auto_now_add=True)

   # user = models.ForeignKey(BaseUser, on_delete=models.CASCADE)
   user_content_type = models.ForeignKey(ContentType, related_name="comment_user", on_delete=models.CASCADE)
   user_object_id = models.PositiveIntegerField()
   user = GenericForeignKey("user_content_type", "user_object_id")

   class Meta:
      abstract = True
      ordering = ['-timestamp']
      indexes = [
         models.Index(fields=["user_content_type", "user_object_id"]),
      ]
```



```python
class Chat (Comment):
   user_content_type = models.ForeignKey(ContentType, related_name="chat_user", on_delete=models.CASCADE)

   application = models.ForeignKey(Application, on_delete=models.CASCADE)

class Review (Comment):
   user_content_type = models.ForeignKey(ContentType, related_name="review_user", on_delete=models.CASCADE)

   shelter = models.ForeignKey(PetShelter, related_name="review_shelter", on_delete=models.CASCADE)
   rating = models.IntegerField(blank=True, null=True)
```

Account
Endpoint: api/account/api/register/shelter/
Methods: POST, , CREATE
Payload: username, email, password, phoneNumber, location, missionStatement, totalRating, numberOfRating
Description: registers the user as a pet shelter

Endpoint: api/account/api/register/seeker/
Methods: POST, CREATE
Payload: username, email, password, phoneNumber, location, preference
Description: registers the user as a pet seeker

Endpoint: api/account/api/login/
Methods: POST
Payload: email, password
Description: logs in the user, seeker or shelter, using email and password

Endpoint: api/account/api/shelter/<int:id>/
Methods: GET, DELETE
Payload: id, username, email, password, phoneNumber, location, missionStatement, totalRating, numberOfRating
Description: returns the info of a specific shelter based on their id. Everyone can view a shelter profile. The same shelter can also delete their profile/user in this page

Endpoint: api/account/api/seeker/<int:id>/
Methods: GET, DELETE
Payload: id, username, email, password, phoneNumber, location, preference
Description: returns the info of a specific seeker based on their id, if it is the same seeker, or a shelter who has an active application with said seeker. Different seekers cannot view other seeker's profiles. The same seeker can also delete their profile/user in this page

Endpoint: api/account/api/shelter/<int:id>/update/
Methods: GET, PATCH
Payload: id, username, email, password, phoneNumber, location, missionStatement, totalRating, numberOfRating
Description: update a specific shelter based on their id.

Endpoint: api/account/api/seeker/<int:id>/update/
Methods: GET, PATCH
Payload: id, username, email, password, phoneNumber, location, preference
Description: update a specific seeker based on their id.

Endpoint: api/account/api/all/shelter
Methods: GET
Payload: id, username, email, password, phoneNumber, location, missionStatement, totalRating, numberOfRating
Description: shows all users every shelter profile, and their info

Pet

Endpoint: api/pet/
Methods: POST, CREATE
Payload: name, status, photo, shelter, description, behavior, medicalHistory, specialNeeds, age, breed, gender, size, species, color, location, fee
Description: creating a pet that belongs to the shelter that is logged in

Endpoint: /api/pet/<int:pet_id>/
Methods: DELETE, PATCH, GET
Payload: name, status, photo, shelter, description, behavior, medicalHistory, specialNeeds, age, breed, gender, size, species, color, location, fee

Description: allows the shelter to view a specific pet based on their id. Allows shelters to delete, or update info said pet

Endpoint: /api/pet/list/
Methods: GET
Payload: pet, name, age, size, species, status
Description: a search list with all pets that belong to a specific shelter. Status being available is automatic

Application

Endpoint: /api/application/
Methods: POST
Payload: reason, seeker, pet
Description: adds an application to an existing “available” posting

Endpoint: api/application/<int:application_id>/
Methods: PATCH
Payload: status
Description: updates an existing application’s status

Endpoint: api/application/list/
Methods: GET
Payload: status, reason, seeker, pet, timestamp, last_updated
Description: view a list of all the applications for a given shelter

Comment

Endpoint: api/chat/<int:application_id>/
Methods: POST, CREATE
Payload: user_content_type, application, content, timestamp
Description: creates chat between the respective pet seeker and shelter who shares an active application

Endpoint: api/chat/<int:application_id>/list/
Methods:GET
Payload: user_content_type, application, content, timestamp
Description: Shows the whole chat between shelter and seeker who shares an active application

Endpoint: api/review/<int:shelter_id>/
Methods: POST, CREATE
Payload: user_ content_type, shelter, rating, content, timestamp
Description: creates a single review for a specific shelter

Endpoint: api/review/<int:shelter_id>/list/
Methods: GET
Payload: user_ content_type, shelter, rating, content, timestamp
Description: a list of all reviews for a specific shelter

Notification

Endpoint: api/notification/
Methods:
Payload: user_content_type, user_object_id, user, status, forward_content_type, forward_object_id, forward, timestamp, content
Description: creates a notification if an application has been sent from a seeker to a shelter, or if the status of an existing application has been updated, or if a review has been created for a shelter, or if a new chat message is sent between seeker and shelter who share an active application

Endpoint: api/notification/<int:notification_id>/
Methods: GET
Payload: user_content_type, user_object_id, user, status, forward_content_type, forward_object_id, forward, timestamp, content
Description: shows a specific notification_id and the contents of it

Endpoint: api/notification/list/<int:user_id>/
Methods: GET
Payload: user_content_type, user_object_id, user, status, forward_content_type, forward_object_id, forward, timestamp, content
Description: shows a list of notification based of a specific user_id