SUNWAY
INT'L BUSINESS SCHOOL

Programme Name: Bcs.It (Hons) (March intake 2022)

Course Code: CSC1015

Course Name: Applied Programming

Individual_project 01

Date of Submission: 31th jan

**Submitted By:**

Student Name: Satkar Niraula

Intake: March 2022

**Submitted To:**

Faculty Name: Umesh Regmi

Department: PO

**Documentation:**

**About the Website:**

New_Social_Media is a web platform where user can basically help each other out.

Here, new users can register their and create their account. The users doesn't need to follow each other in order to view and like each others post, it is because I intend to make it so. When a user encounters any kind of problem in any thing they can create a post and any other user who registered for the social media can view it.
The other viewing users can apply for the post and are listed as applicants for the user of the corresponding post.
The user of that post gets the applicants for that post if applied by other users.
The user can approve the applicant and can chat with them about their problem until it becomes solve
The posts can be edited and deleted by the post owner only.

I intend to improve this project, but for the initial phase I have decided to showcase these only

**Google Drive Link:** *https://drive.google.com/file/d/1B0WH95pteI4xl3q08gHwU3CbgYvcjrNF/view?usp=sharing*

**Key_Features:**

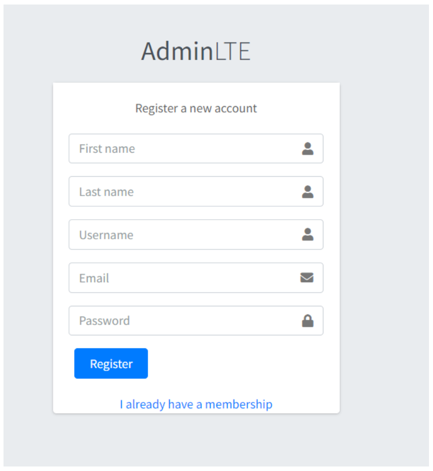
1. User registration/ User login/ logout
2. Post creation – edit/delete
3. apply to the post
4. approve the applicants
5. chat with the applicants and vice-versa

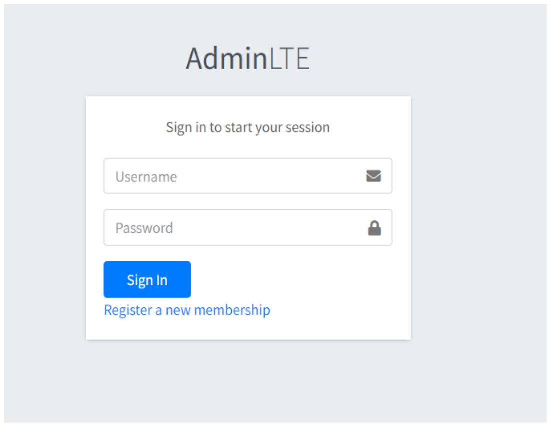
**Running the program:**

1. Should create a new environment
2. Install the requirements.txt text within the environment
3. Activate the created environment
4. Create your own migrations and migrate
5. Run the server

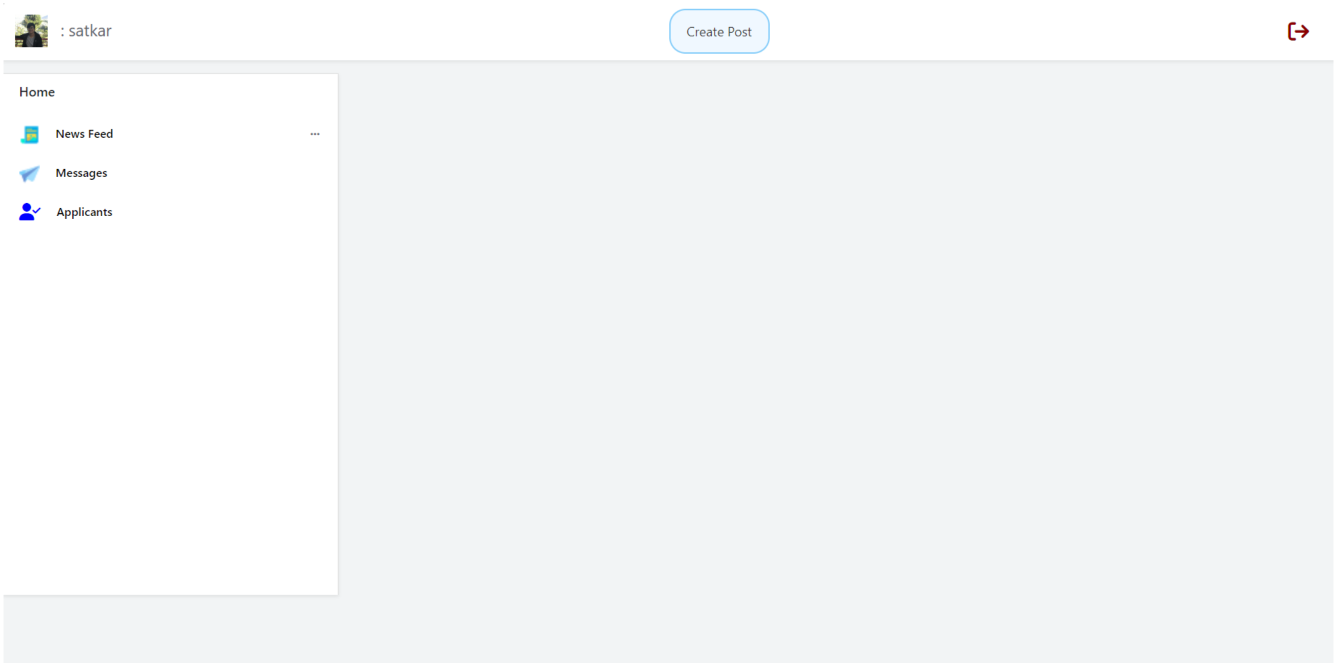
## Web Application Screenshots:

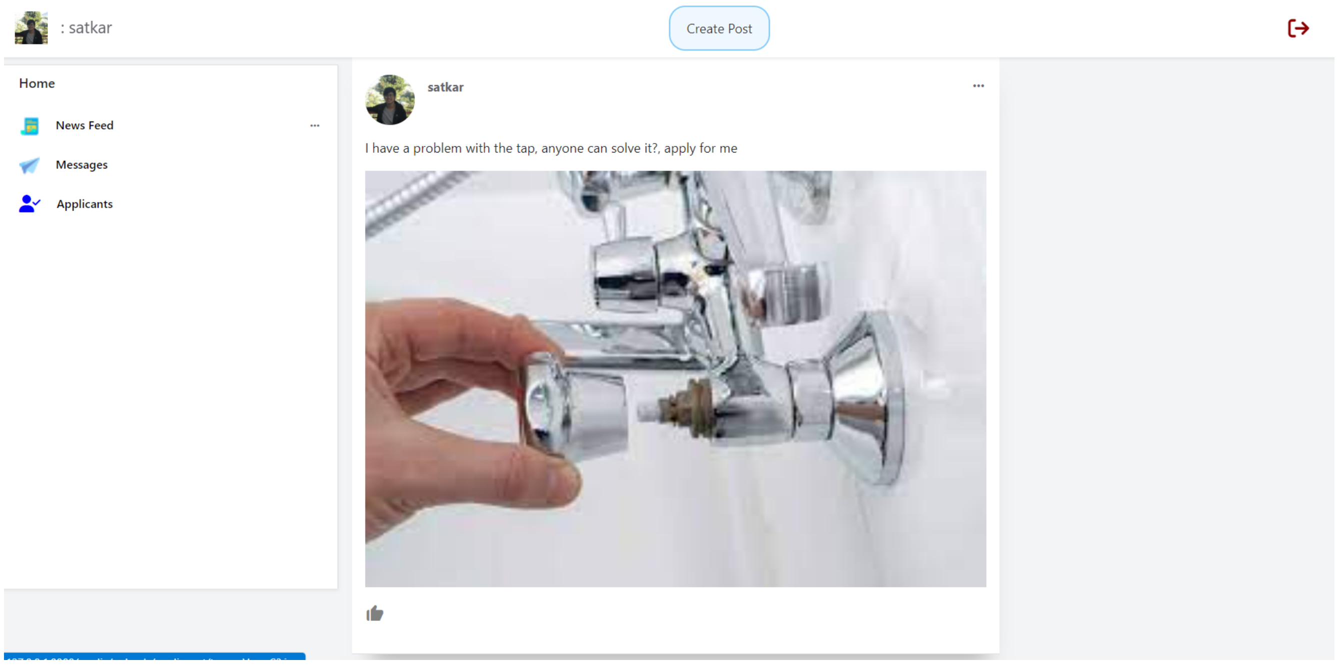
## 1. User Registration/login



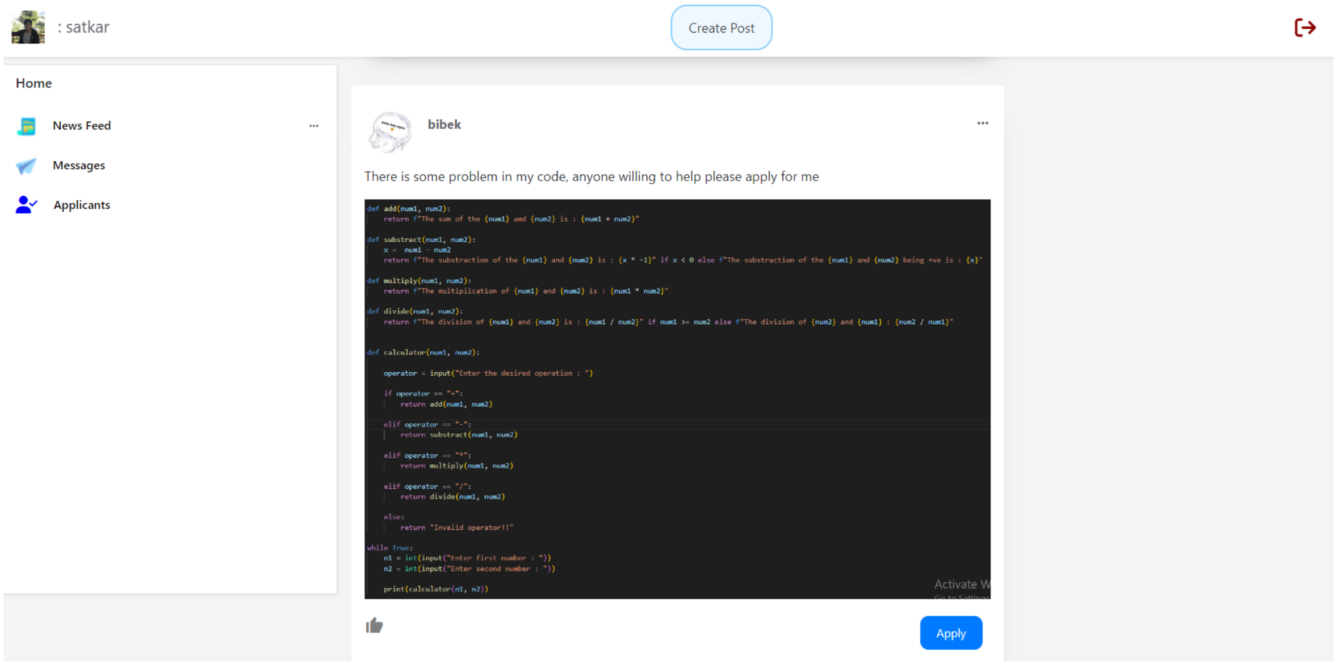


## 2. UserFeed/Dashboard:





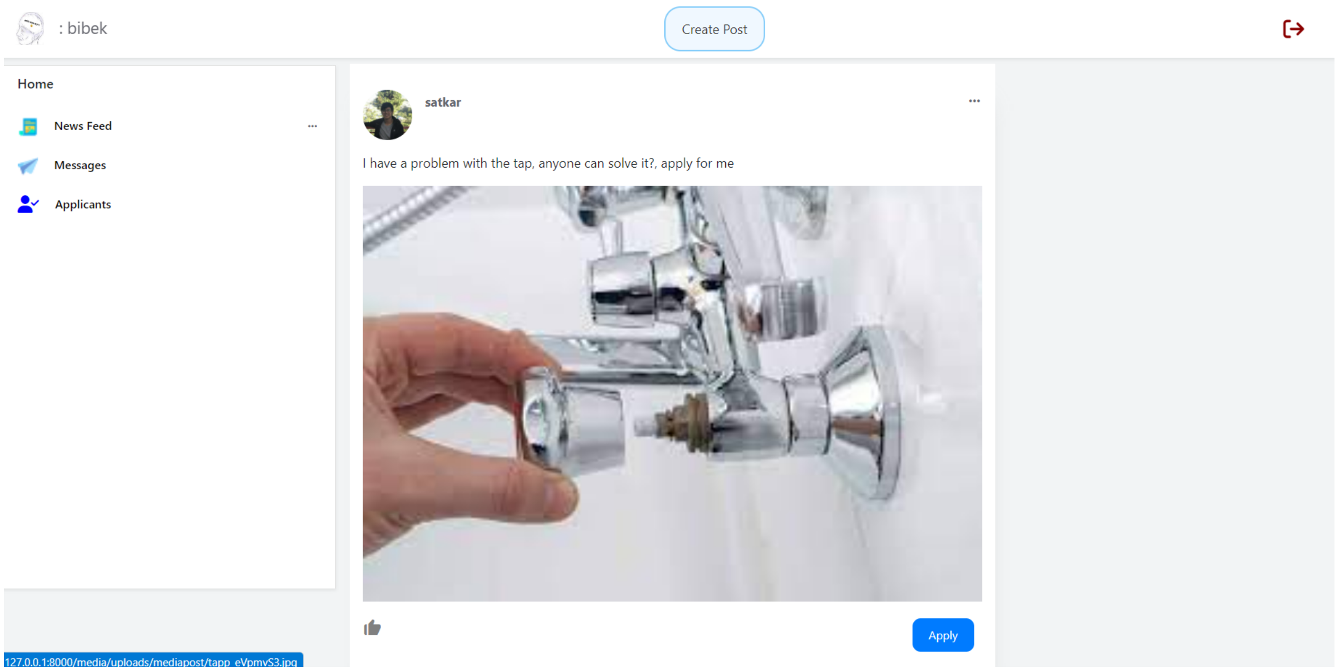
I have a problem with the tap, anyone can solve it?, apply for me



There is some problem in my code, anyone willing to help please apply for me

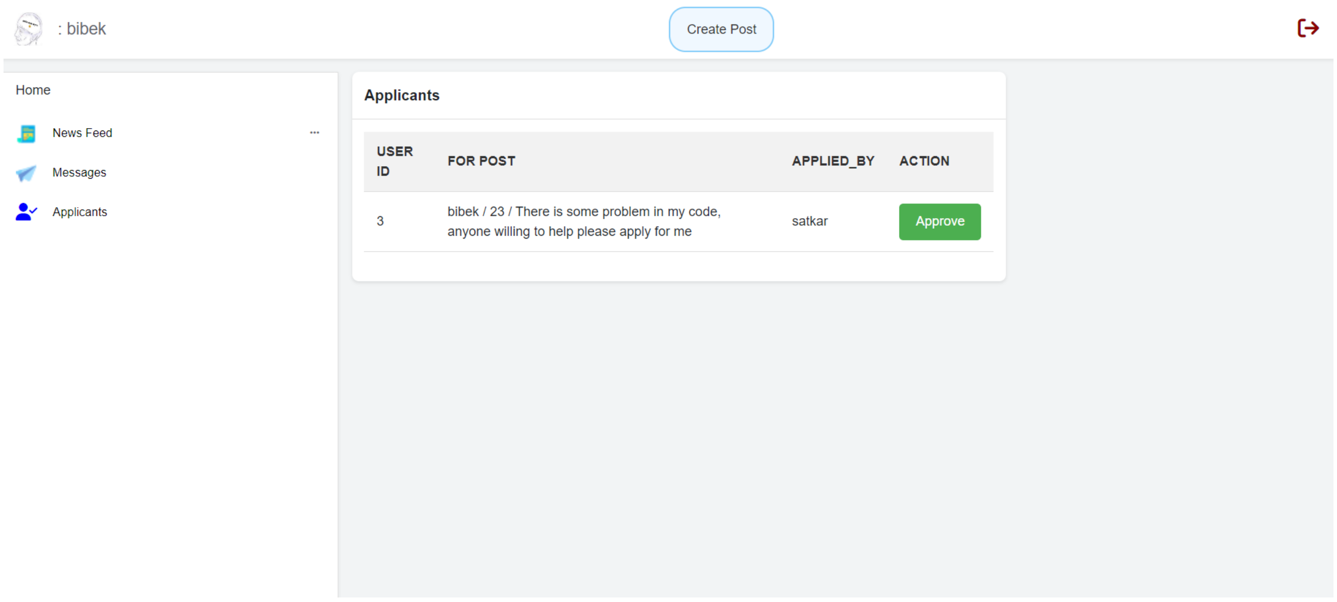
```python
def add(num1, num2):
    return f"The sum of the {num1} amd {num2} is : {num1 + num2}"

def substract(num1, num2):
    x =  num1 - num2
    return f"The substraction of the {num1} and {num2} is : {x * -1}" if x < 0 else f"The substraction of the {num1} and {num2} being +ve is : {x}"

def multiply(num1, num2):
    return f"The multiplication of {num1} and {num2} is : {num1 * num2}"

def divide(num1, num2):
    return f"The division of {num1} and {num2} is : {num1 / num2}" if num1 >= num2 else f"The division of {num2} and {num1} : {num2 / num1}"


def calculator(num1, num2):

    operator = input("Enter the desired operation : ")

    if operator == "+":
        return add(num1, num2)

    elif operator == "-":
        return substract(num1, num2)

    elif operator == "*":
        return multiply(num1, num2)

    elif operator == "/":
        return divide(num1, num2)

    else:
        return "Invalid operator!!"

while True:
    n1 = int(input("Enter first number : "))
    n2 = int(input("Enter second number : "))

    print(calculator(n1, n2))
```

This userFeed comes when we click the NewsFeed
We can like and unlike the posts and we can only apply for other user’s posts



I have a problem with the tap, anyone can solve it?, apply for me

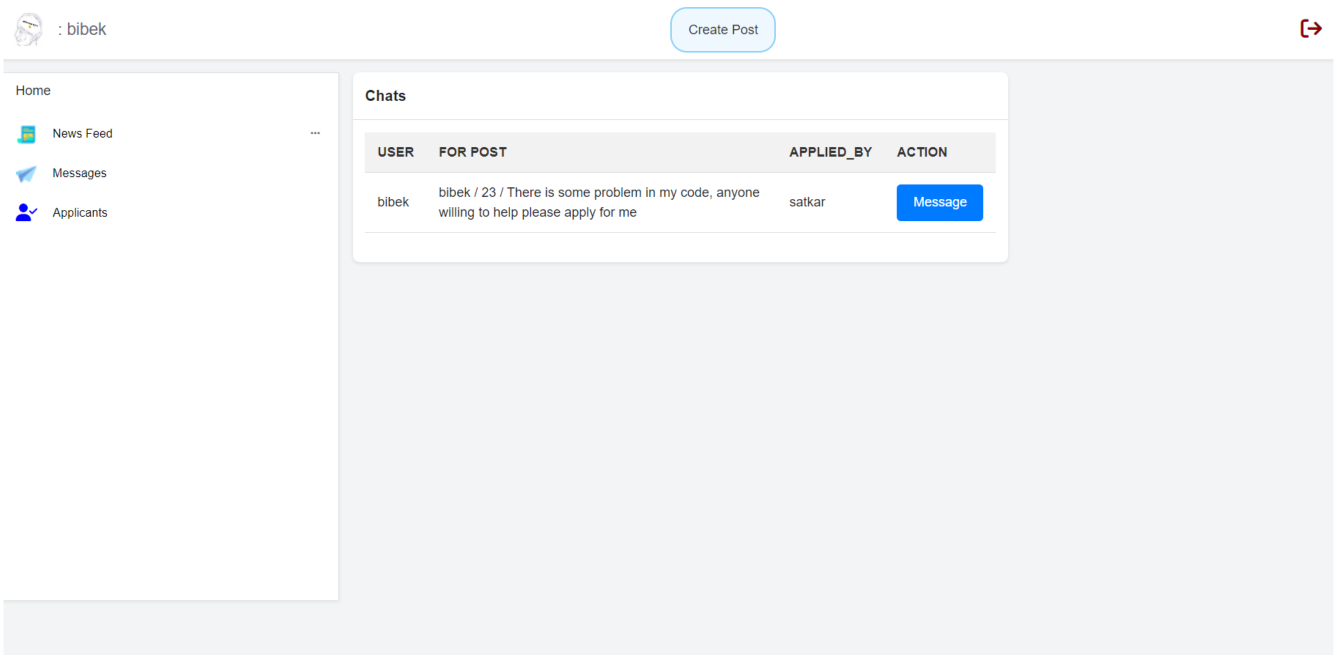
Now in the applicants of bibek in which I applied from Satkar in the post of bibek,



Applicants

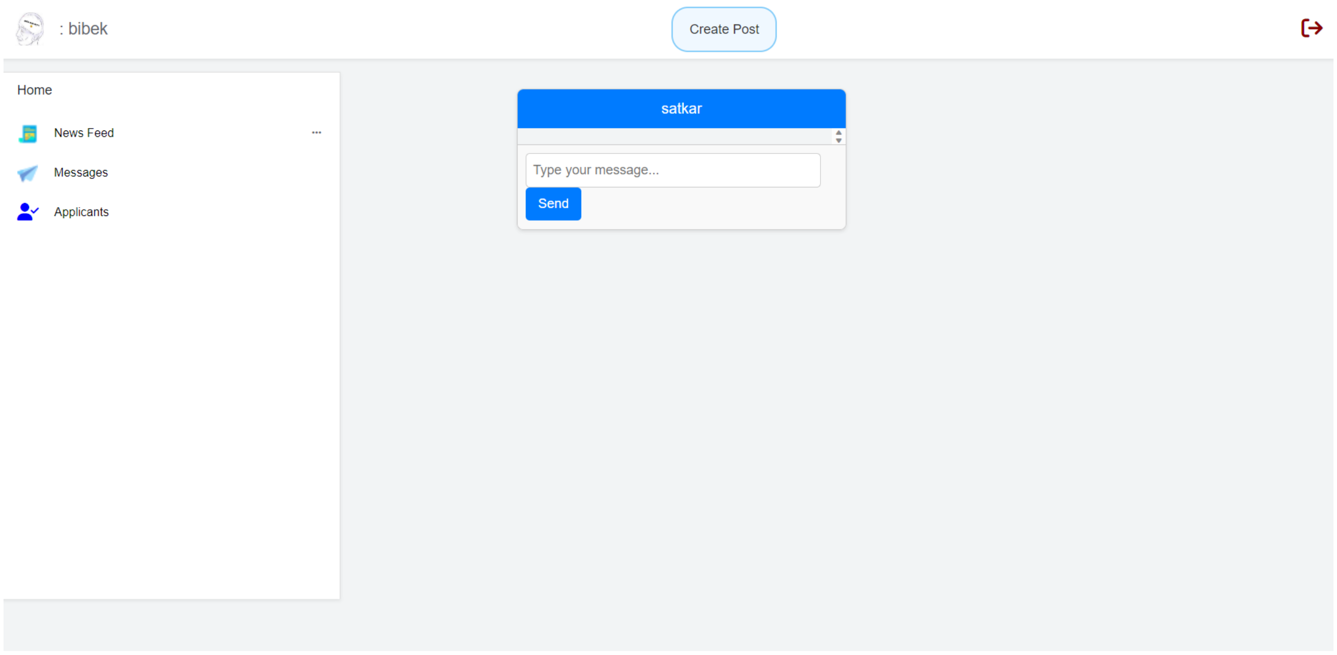
| USER ID | FOR POST | APPLIED_BY | ACTION |
|---|---|---|---|
| 3 | bibek / 23 / There is some problem in my code, anyone willing to help please apply for me | satkar | Approve |

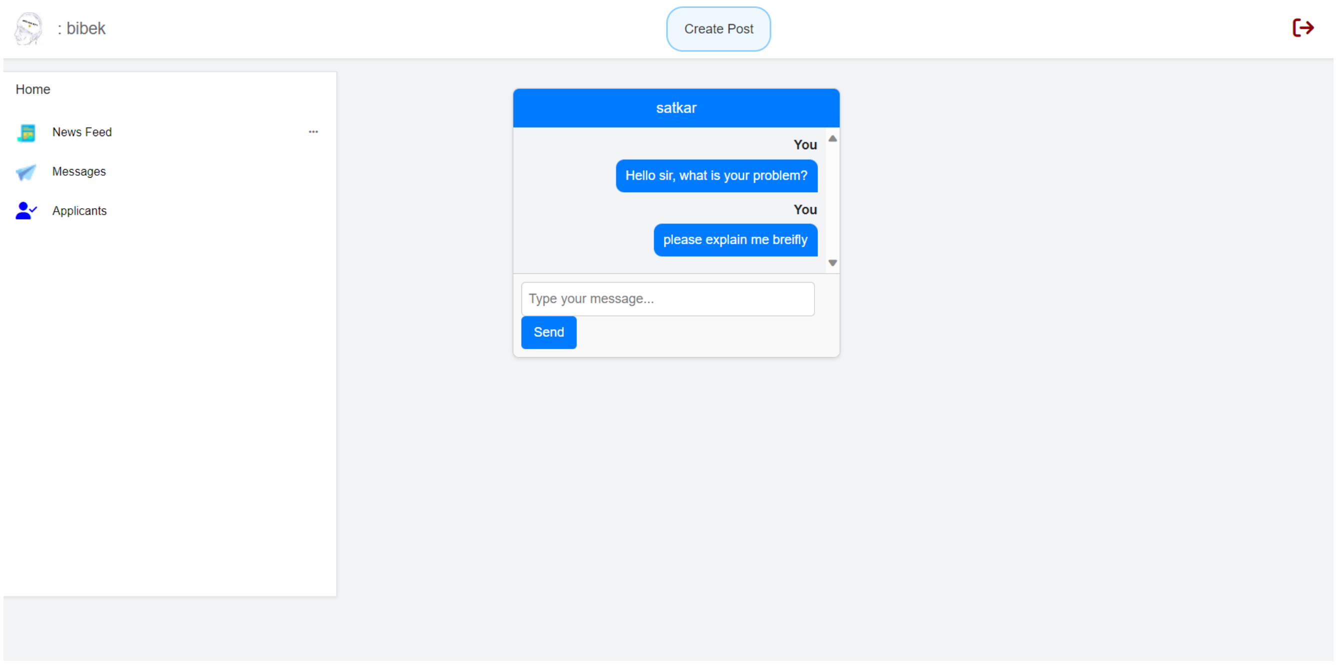
After we approve it redirects to the messeges and create a new conversatin between that post owner and the applicant of the post which is satkar for bibek,



| USER | FOR POST | APPLIED_BY | ACTION |
|---|---|---|---|
| bibek | bibek / 23 / There is some problem in my code, anyone willing to help please apply for me | satkar | Message |

we click the message button to chat within the created conversation,

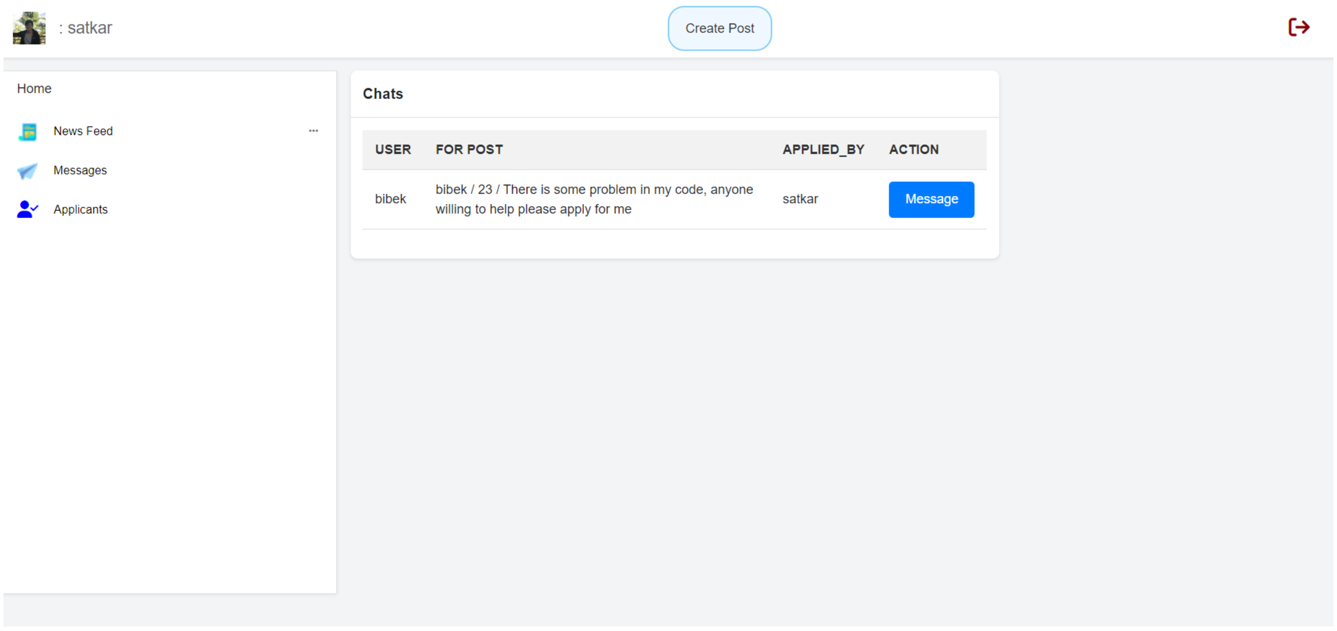




Hello sir, what is your problem?

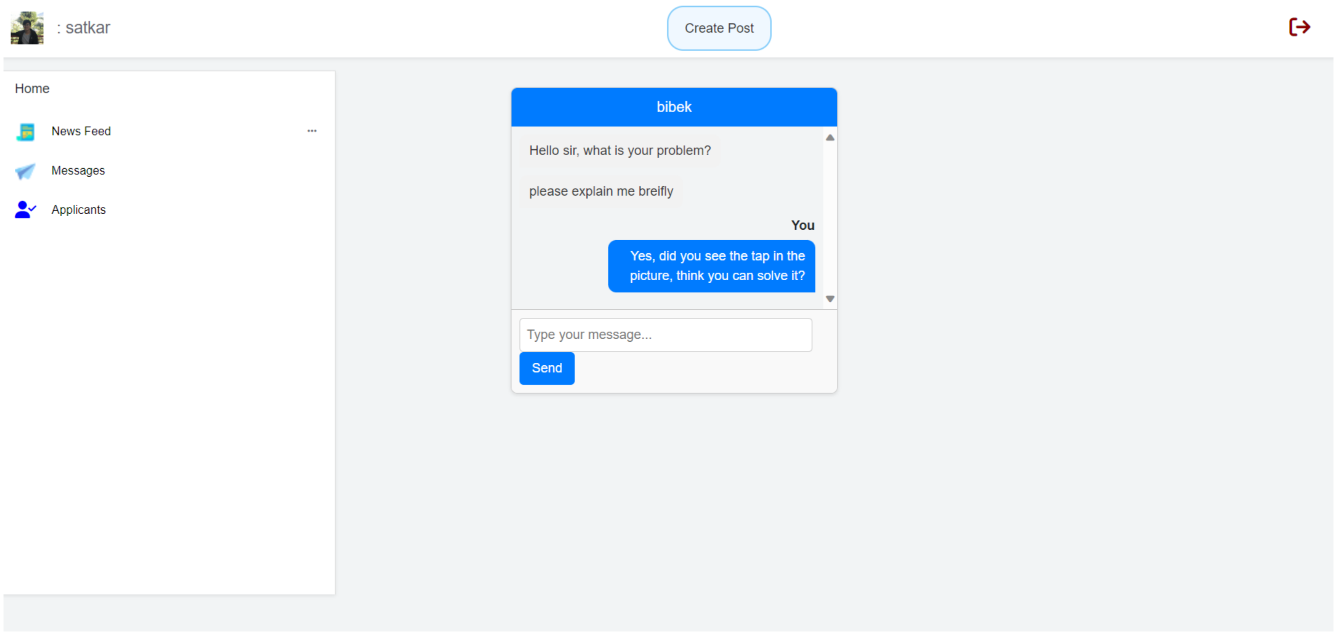
please explain me breifly

And we can find this chat with the applicant’s account too,



Chats

| USER | FOR POST | APPLIED_BY | ACTION |
|---|---|---|---|
| bibek | bibek / 23 / There is some problem in my code, anyone willing to help please apply for me | satkar | Message |



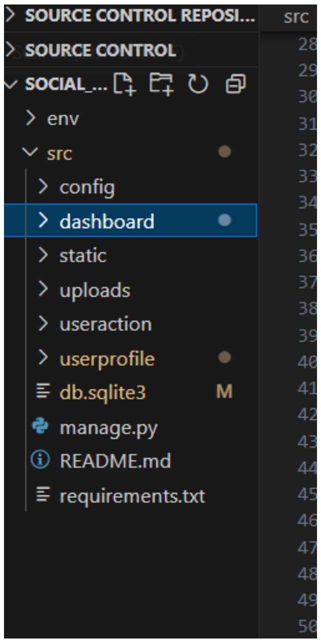
Hello sir, what is your problem?

please explain me breifly

You

Yes, did you see the tap in the picture, think you can solve it?

## Backend:



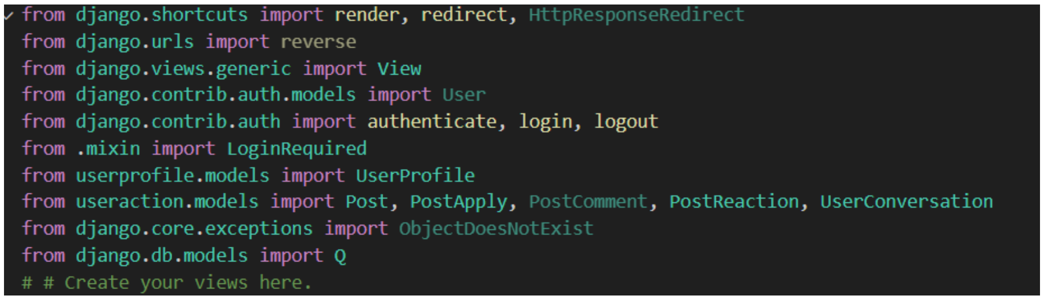
```
env
src
  config
  dashboard
  static
  uploads
  useraction
  userprofile
  db.sqlite3      M
  manage.py
  README.md
  requirements.txt
```

In this directory view, we can see the dashboard app, useaction app for users' activity with the social media and related model are there

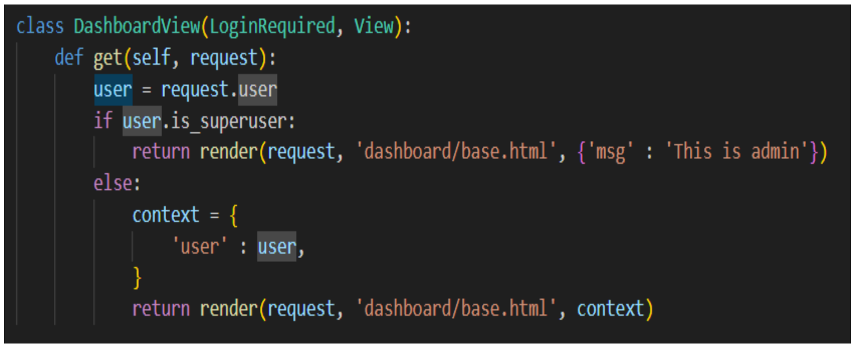
In the userprofile app there is model for user creation and authentication activity

## 1. CRUD functinality views:

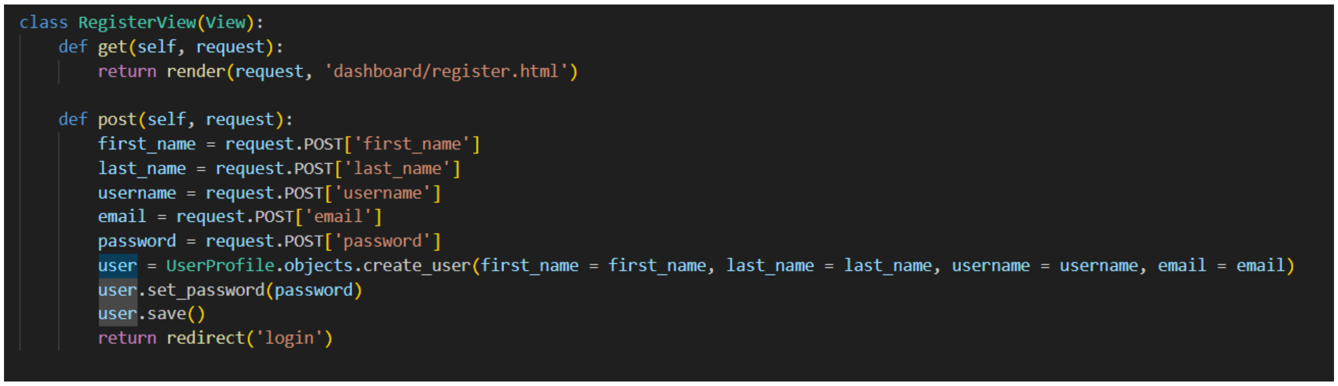
### User Registration/Login/Logout:



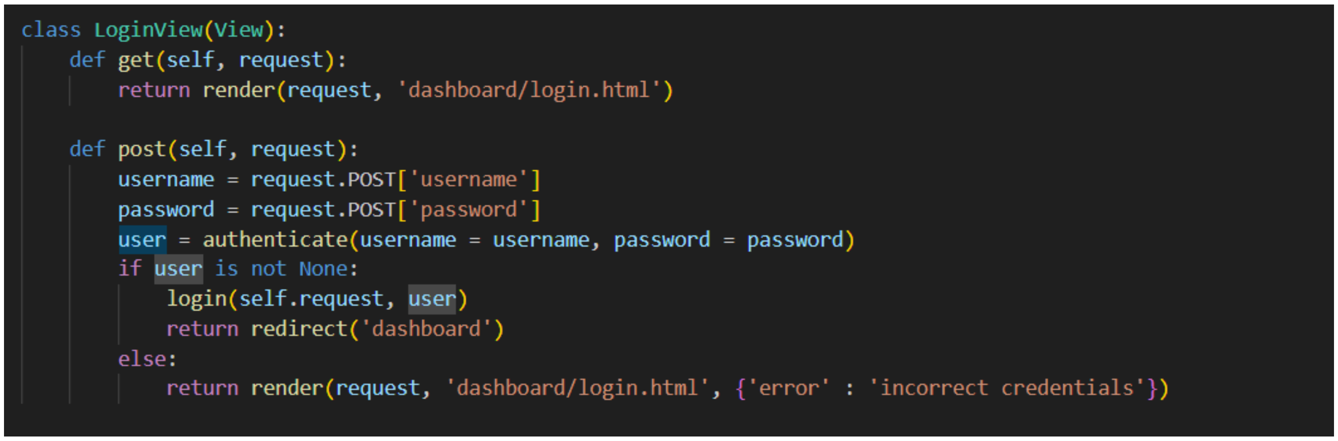
```python
from django.shortcuts import render, redirect, HttpResponseRedirect
from django.urls import reverse
from django.views.generic import View
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from .mixin import LoginRequired
from userprofile.models import UserProfile
from useraction.models import Post, PostApply, PostComment, PostReaction, UserConversation
from django.core.exceptions import ObjectDoesNotExist
from django.db.models import Q
# # Create your views here.
```



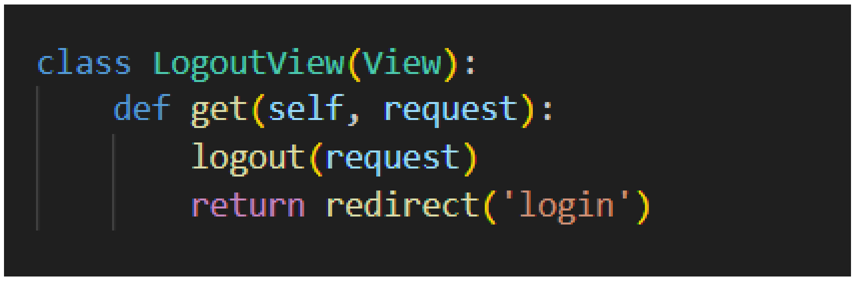
```python
class DashboardView(LoginRequired, View):
    def get(self, request):
        user = request.user
        if user.is_superuser:
            return render(request, 'dashboard/base.html', {'msg' : 'This is admin'})
        else:
            context = {
                'user' : user,
            }
            return render(request, 'dashboard/base.html', context)
```



```python
class RegisterView(View):
    def get(self, request):
        return render(request, 'dashboard/register.html')

    def post(self, request):
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        user = UserProfile.objects.create_user(first_name = first_name, last_name = last_name, username = username, email = email)
        user.set_password(password)
        user.save()
        return redirect('login')
```

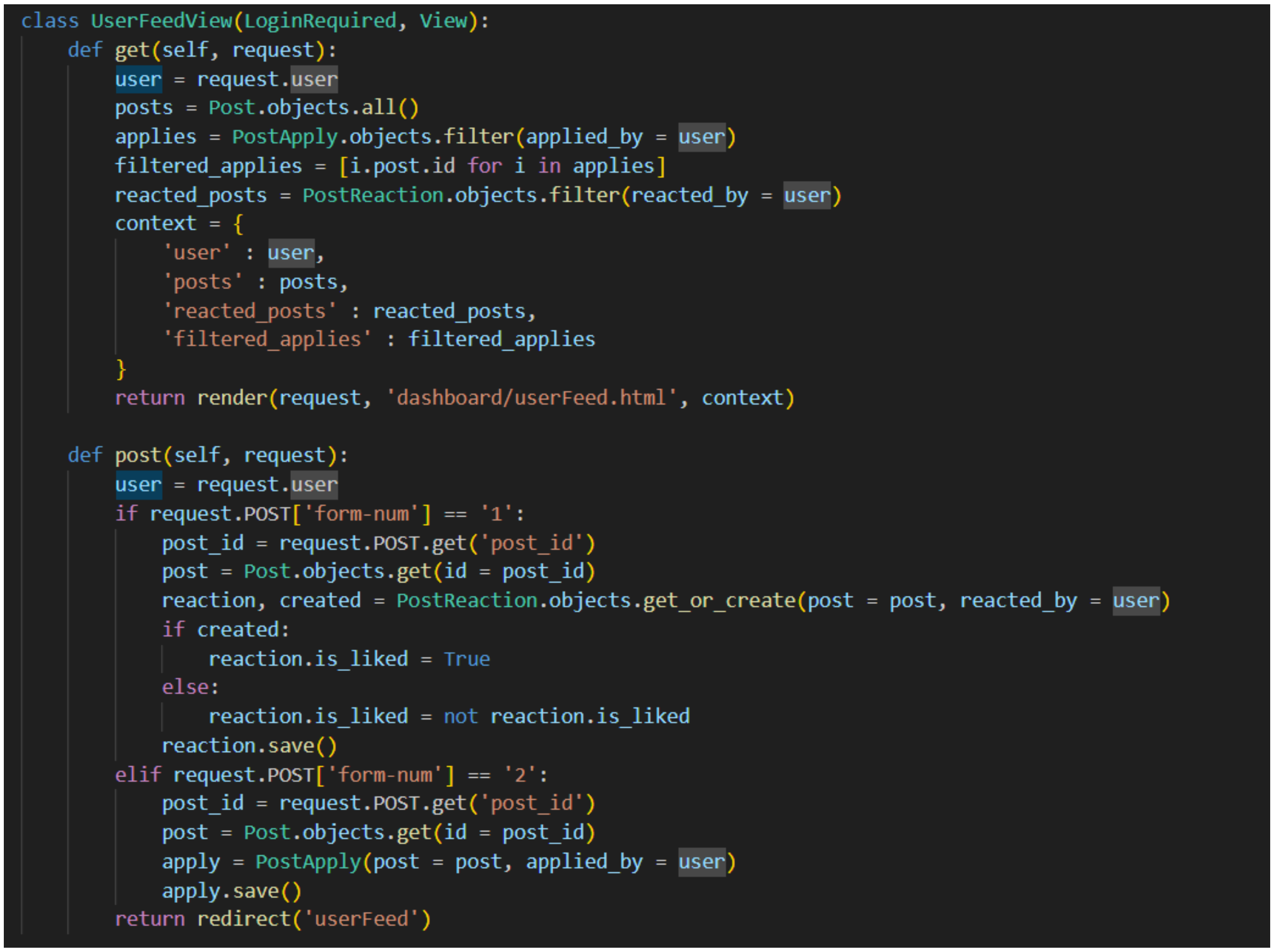


```python
class LoginView(View):
    def get(self, request):
        return render(request, 'dashboard/login.html')

    def post(self, request):
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username = username, password = password)
        if user is not None:
            login(self.request, user)
            return redirect('dashboard')
        else:
            return render(request, 'dashboard/login.html', {'error' : 'incorrect credentials'})
```



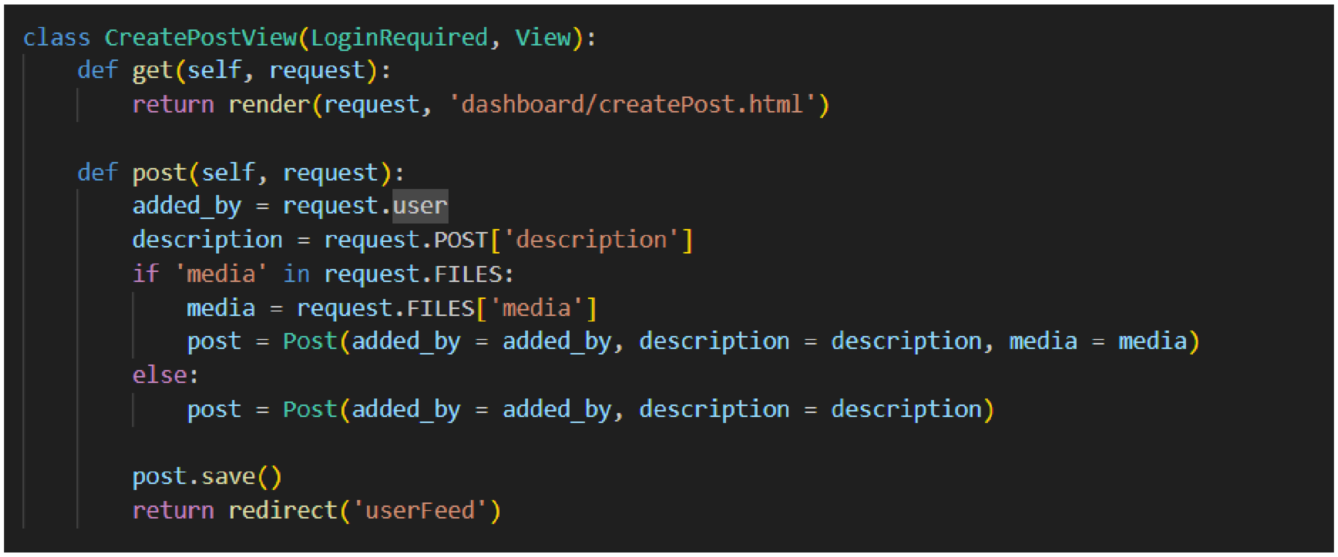
```python
class LogoutView(View):
    def get(self, request):
        logout(request)
        return redirect('login')
```

**For Viewing the social media users' posts and like/unlike and applicants creations:**

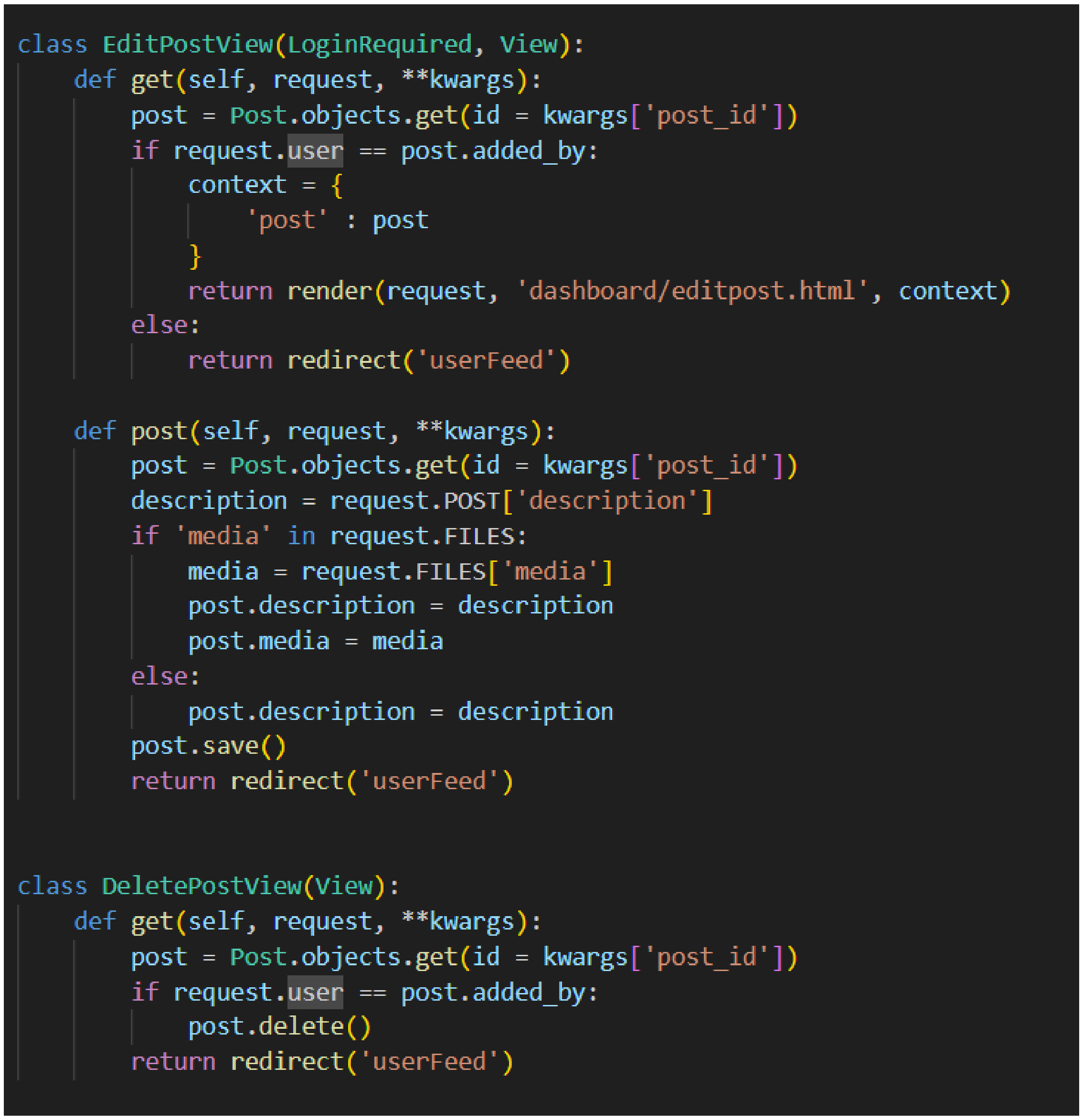


```python
class UserFeedView(LoginRequired, View):
    def get(self, request):
        user = request.user
        posts = Post.objects.all()
        applies = PostApply.objects.filter(applied_by = user)
        filtered_applies = [i.post.id for i in applies]
        reacted_posts = PostReaction.objects.filter(reacted_by = user)
        context = {
            'user' : user,
            'posts' : posts,
            'reacted_posts' : reacted_posts,
            'filtered_applies' : filtered_applies
        }
        return render(request, 'dashboard/userFeed.html', context)

    def post(self, request):
        user = request.user
        if request.POST['form-num'] == '1':
            post_id = request.POST.get('post_id')
            post = Post.objects.get(id = post_id)
            reaction, created = PostReaction.objects.get_or_create(post = post, reacted_by = user)
            if created:
                reaction.is_liked = True
            else:
                reaction.is_liked = not reaction.is_liked
            reaction.save()
        elif request.POST['form-num'] == '2':
            post_id = request.POST.get('post_id')
            post = Post.objects.get(id = post_id)
            apply = PostApply(post = post, applied_by = user)
            apply.save()
        return redirect('userFeed')
```

**Creating New post, editing deleting:**

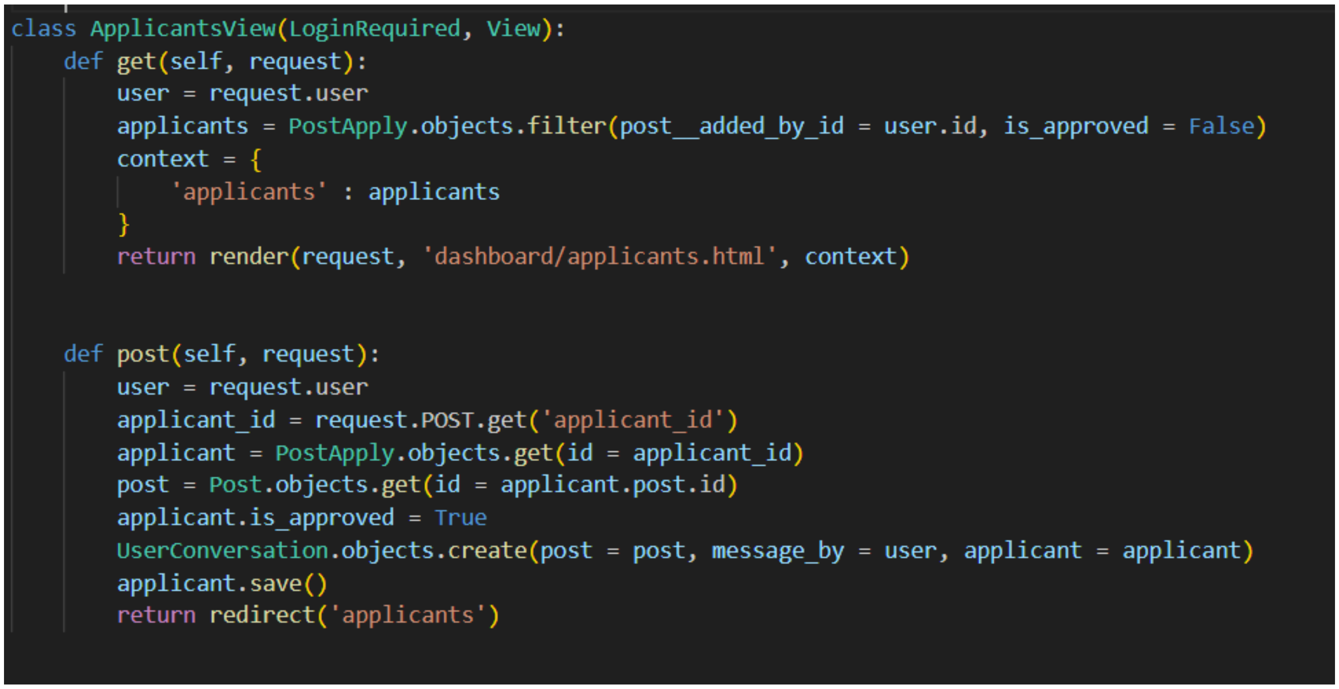


```python
class CreatePostView(LoginRequired, View):
    def get(self, request):
        return render(request, 'dashboard/createPost.html')

    def post(self, request):
        added_by = request.user
        description = request.POST['description']
        if 'media' in request.FILES:
            media = request.FILES['media']
            post = Post(added_by = added_by, description = description, media = media)
        else:
            post = Post(added_by = added_by, description = description)

        post.save()
        return redirect('userFeed')
```



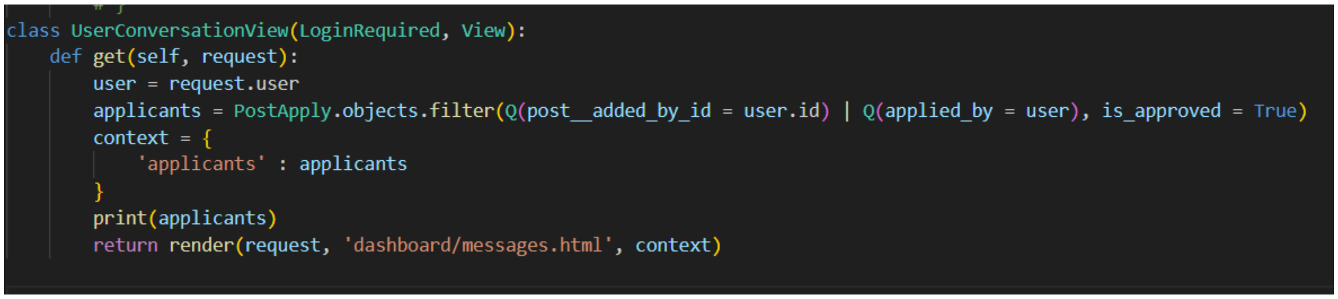
```python
class EditPostView(LoginRequired, View):
    def get(self, request, **kwargs):
        post = Post.objects.get(id = kwargs['post_id'])
        if request.user == post.added_by:
            context = {
                'post' : post
            }
            return render(request, 'dashboard/editpost.html', context)
        else:
            return redirect('userFeed')

    def post(self, request, **kwargs):
        post = Post.objects.get(id = kwargs['post_id'])
        description = request.POST['description']
        if 'media' in request.FILES:
            media = request.FILES['media']
            post.description = description
            post.media = media
        else:
            post.description = description
        post.save()
        return redirect('userFeed')


class DeletePostView(View):
    def get(self, request, **kwargs):
        post = Post.objects.get(id = kwargs['post_id'])
        if request.user == post.added_by:
            post.delete()
        return redirect('userFeed')
```

**Viewing and approving applicants as well as creating new userconversation for chatting:**



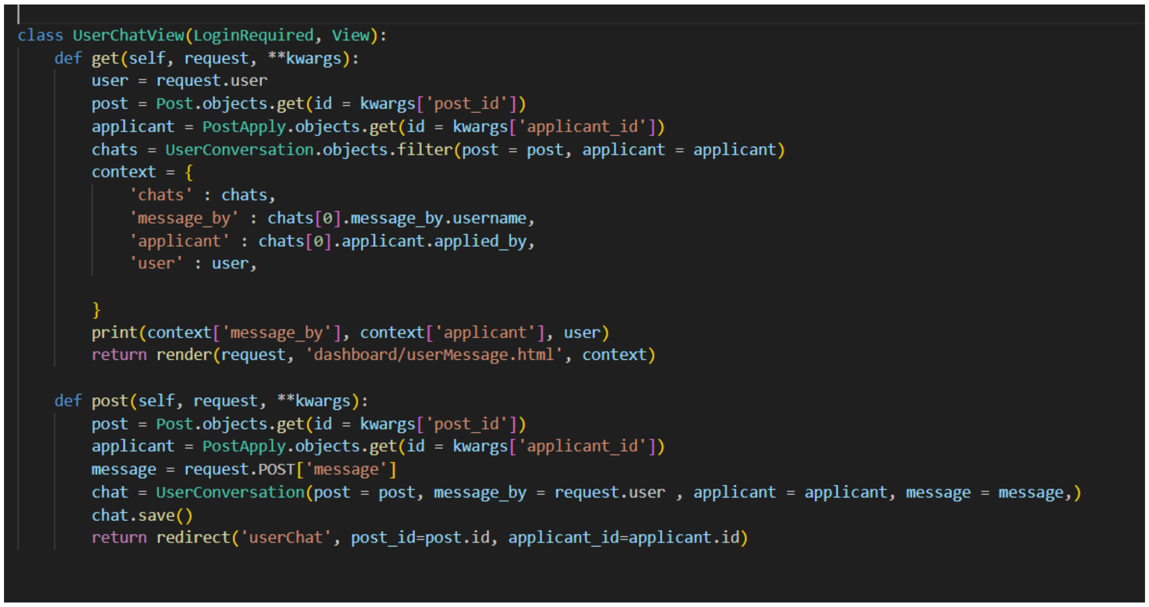
```
class ApplicantsView(LoginRequired, View):
    def get(self, request):
        user = request.user
        applicants = PostApply.objects.filter(post__added_by_id = user.id, is_approved = False)
        context = {
            'applicants' : applicants
        }
        return render(request, 'dashboard/applicants.html', context)


    def post(self, request):
        user = request.user
        applicant_id = request.POST.get('applicant_id')
        applicant = PostApply.objects.get(id = applicant_id)
        post = Post.objects.get(id = applicant.post.id)
        applicant.is_approved = True
        UserConversation.objects.create(post = post, message_by = user, applicant = applicant)
        applicant.save()
        return redirect('applicants')
```

**Viewing the newly created user conversation or chat:**



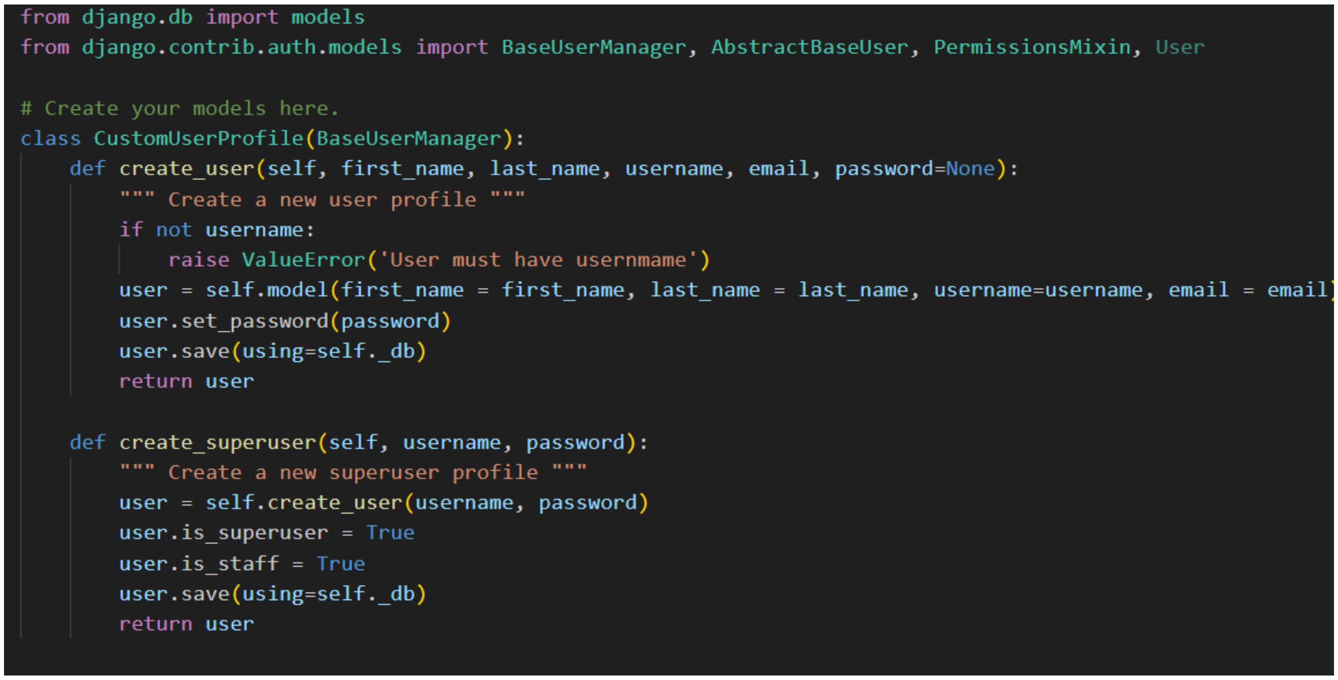
```
class UserConversationView(LoginRequired, View):
    def get(self, request):
        user = request.user
        applicants = PostApply.objects.filter(Q(post__added_by_id = user.id) | Q(applied_by = user), is_approved = True)
        context = {
            'applicants' : applicants
        }
        print(applicants)
        return render(request, 'dashboard/messages.html', context)
```

**Opening the chatbox and chatting:**

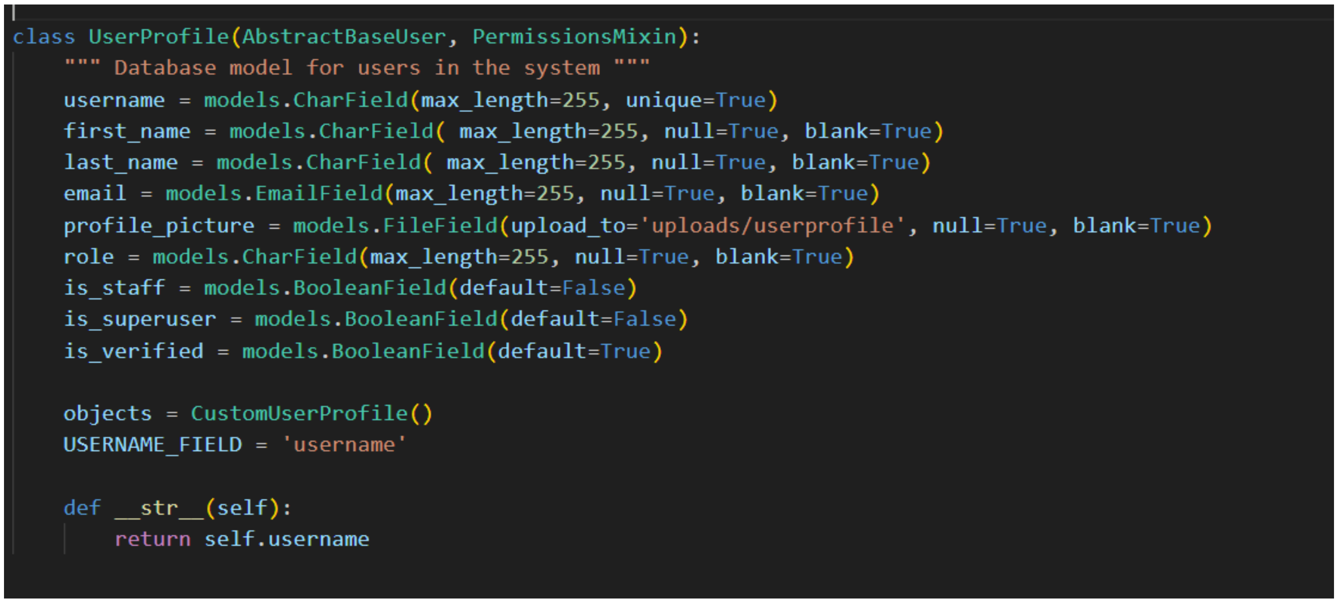


```
class UserChatView(LoginRequired, View):
    def get(self, request, **kwargs):
        user = request.user
        post = Post.objects.get(id = kwargs['post_id'])
        applicant = PostApply.objects.get(id = kwargs['applicant_id'])
        chats = UserConversation.objects.filter(post = post, applicant = applicant)
        context = {
            'chats' : chats,
            'message_by' : chats[0].message_by.username,
            'applicant' : chats[0].applicant.applied_by,
            'user' : user,

        }
        print(context['message_by'], context['applicant'], user)
        return render(request, 'dashboard/userMessage.html', context)

    def post(self, request, **kwargs):
        post = Post.objects.get(id = kwargs['post_id'])
        applicant = PostApply.objects.get(id = kwargs['applicant_id'])
        message = request.POST['message']
        chat = UserConversation(post = post, message_by = request.user , applicant = applicant, message = message,)
        chat.save()
        return redirect('userChat', post_id=post.id, applicant_id=applicant.id)
```

**Model of user for used for registration, login:**
I made a custom user model by over riding the methods of baseusermanager

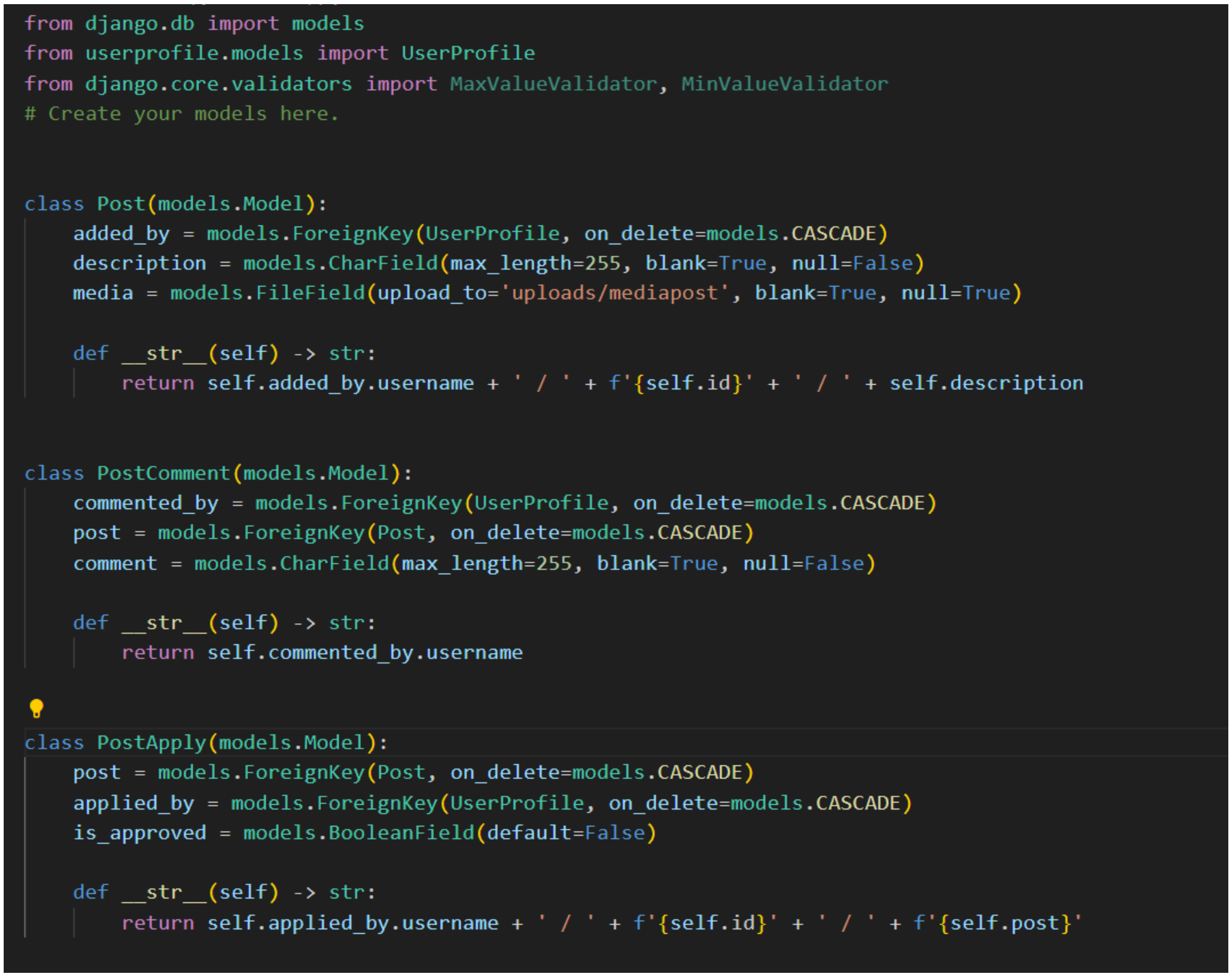


```
from django.db import models
from django.contrib.auth.models import BaseUserManager, AbstractBaseUser, PermissionsMixin, User

# Create your models here.
class CustomUserProfile(BaseUserManager):
    def create_user(self, first_name, last_name, username, email, password=None):
        """ Create a new user profile """
        if not username:
            raise ValueError('User must have usernmame')
        user = self.model(first_name = first_name, last_name = last_name, username=username, email = email
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, username, password):
        """ Create a new superuser profile """
        user = self.create_user(username, password)
        user.is_superuser = True
        user.is_staff = True
        user.save(using=self._db)
        return user
```

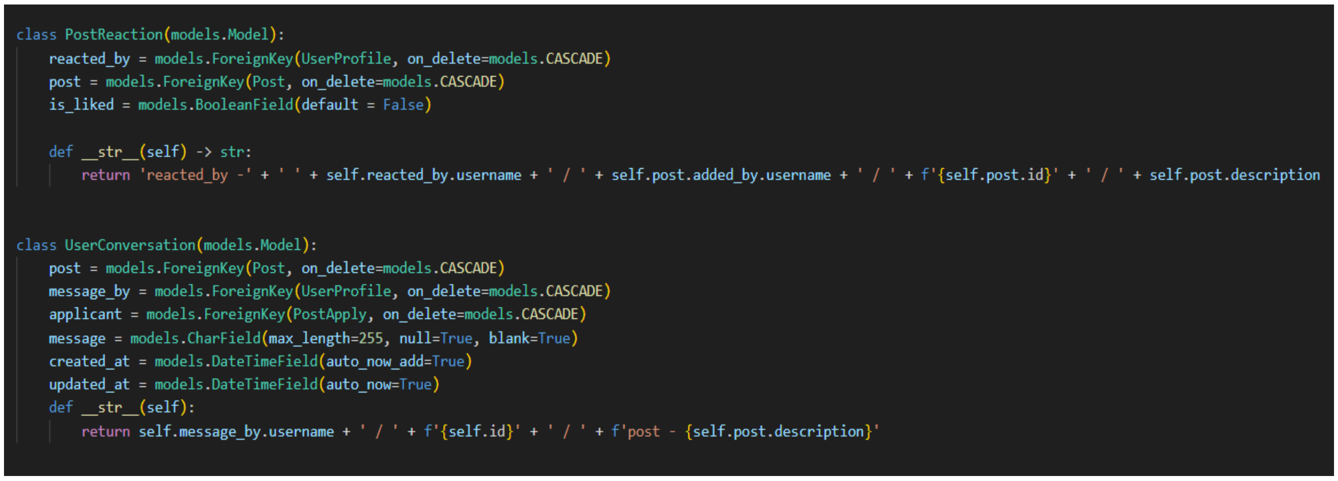


```
class UserProfile(AbstractBaseUser, PermissionsMixin):
    """ Database model for users in the system """
    username = models.CharField(max_length=255, unique=True)
    first_name = models.CharField( max_length=255, null=True, blank=True)
    last_name = models.CharField( max_length=255, null=True, blank=True)
    email = models.EmailField(max_length=255, null=True, blank=True)
    profile_picture = models.FileField(upload_to='uploads/userprofile', null=True, blank=True)
    role = models.CharField(max_length=255, null=True, blank=True)
    is_staff = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)
    is_verified = models.BooleanField(default=True)

    objects = CustomUserProfile()
    USERNAME_FIELD = 'username'

    def __str__(self):
        return self.username
```

**Model for every actions or activity of the user in the web interface:**

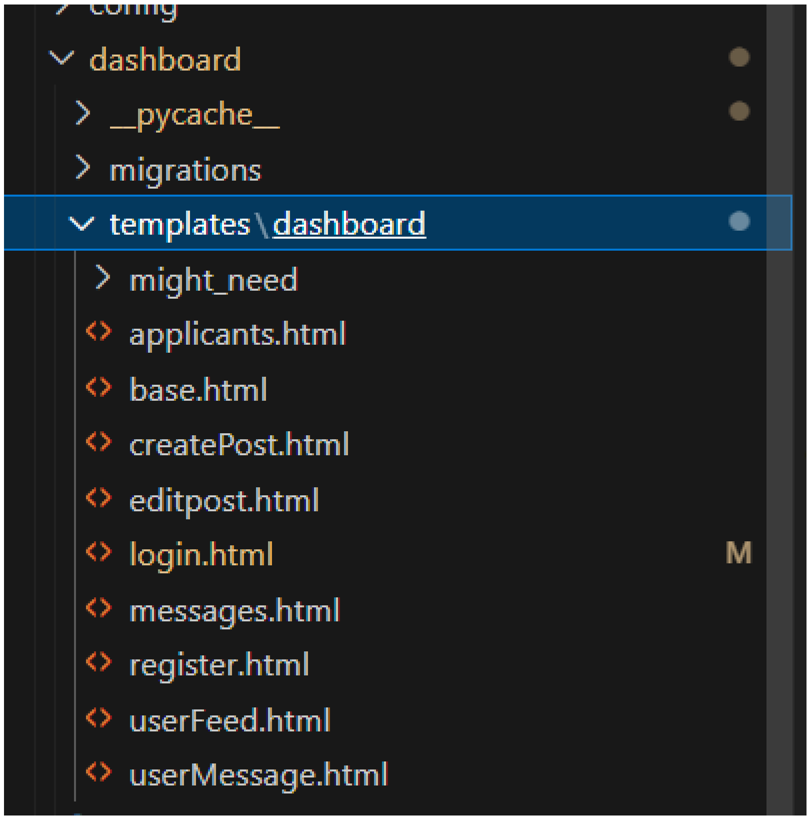


```python
from django.db import models
from userprofile.models import UserProfile
from django.core.validators import MaxValueValidator, MinValueValidator
# Create your models here.


class Post(models.Model):
    added_by = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    description = models.CharField(max_length=255, blank=True, null=False)
    media = models.FileField(upload_to='uploads/mediapost', blank=True, null=True)

    def __str__(self) -> str:
        return self.added_by.username + ' / ' + f'{self.id}' + ' / ' + self.description


class PostComment(models.Model):
    commented_by = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    comment = models.CharField(max_length=255, blank=True, null=False)

    def __str__(self) -> str:
        return self.commented_by.username


class PostApply(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    applied_by = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    is_approved = models.BooleanField(default=False)

    def __str__(self) -> str:
        return self.applied_by.username + ' / ' + f'{self.id}' + ' / ' + f'{self.post}'
```



```python
class PostReaction(models.Model):
    reacted_by = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    is_liked = models.BooleanField(default = False)

    def __str__(self) -> str:
        return 'reacted_by -' + ' ' + self.reacted_by.username + ' / ' + self.post.added_by.username + ' / ' + f'{self.post.id}' + ' / ' + self.post.description


class UserConversation(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    message_by = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    applicant = models.ForeignKey(PostApply, on_delete=models.CASCADE)
    message = models.CharField(max_length=255, null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.message_by.username + ' / ' + f'{self.id}' + ' / ' + f'post - {self.post.description}'
```

**Templates:**



```
> config
v dashboard                  ●
  > __pycache__              ●
  > migrations
  v templates\dashboard      ●
    > might_need
    <> applicants.html
    <> base.html
    <> createPost.html
    <> editpost.html
    <> login.html            M
    <> messages.html
    <> register.html
    <> userFeed.html
    <> userMessage.html
```

**These are the Main highlights and my intention of my project.**