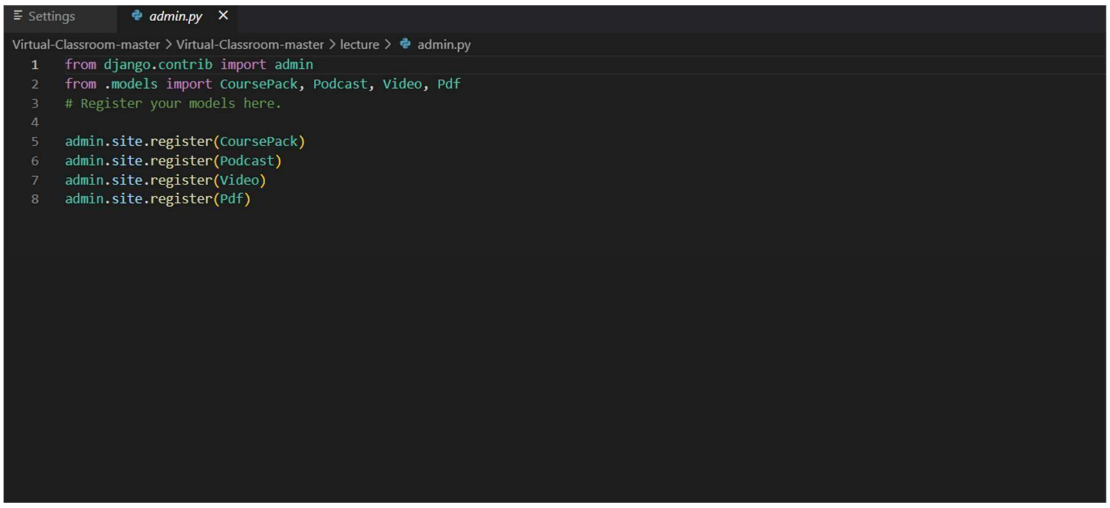
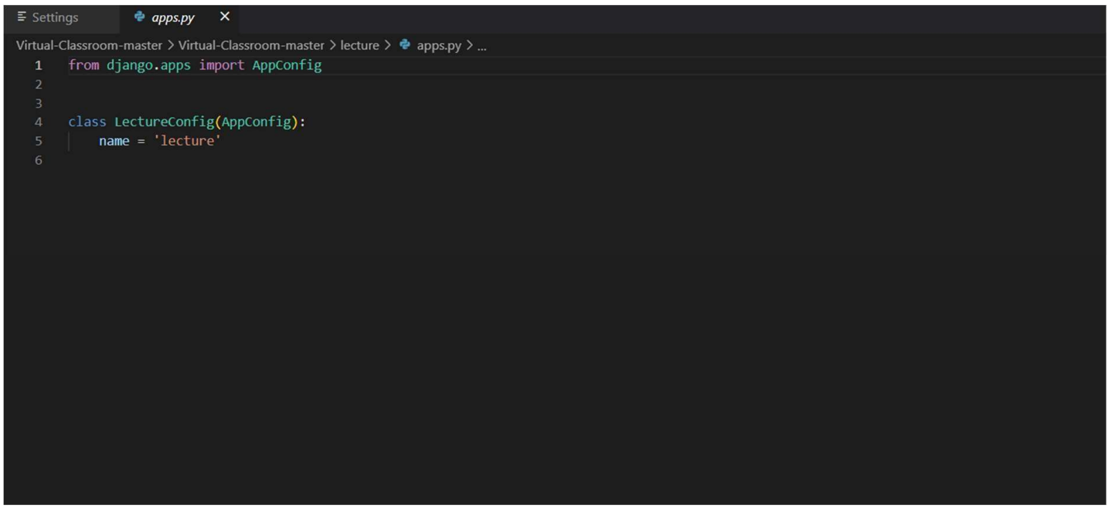
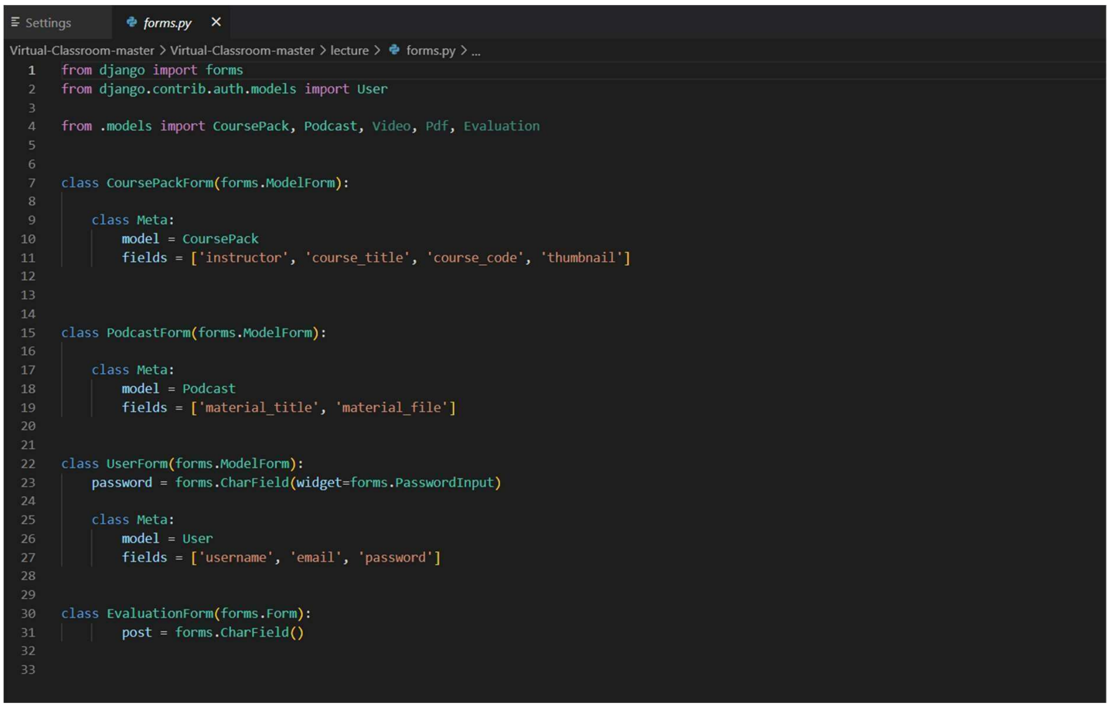
SOURCE CODE:


Settings
admin.py
Virtual-Classroom-master > Virtual-Classroom-master > lecture > admin.py

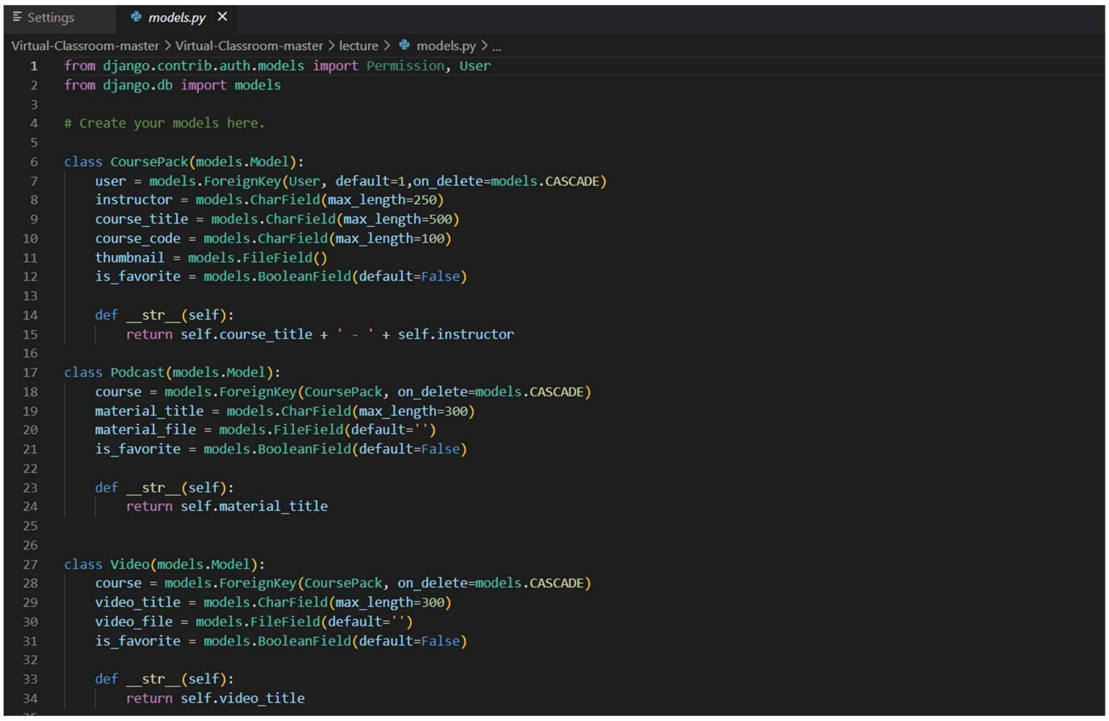
```python
from django.contrib import admin
from .models import CoursePack, Podcast, Video, Pdf
# Register your models here.

admin.site.register(CoursePack)
admin.site.register(Podcast)
admin.site.register(Video)
admin.site.register(Pdf)
```


Settings
apps.py
Virtual-Classroom-master > Virtual-Classroom-master > lecture > apps.py > ...

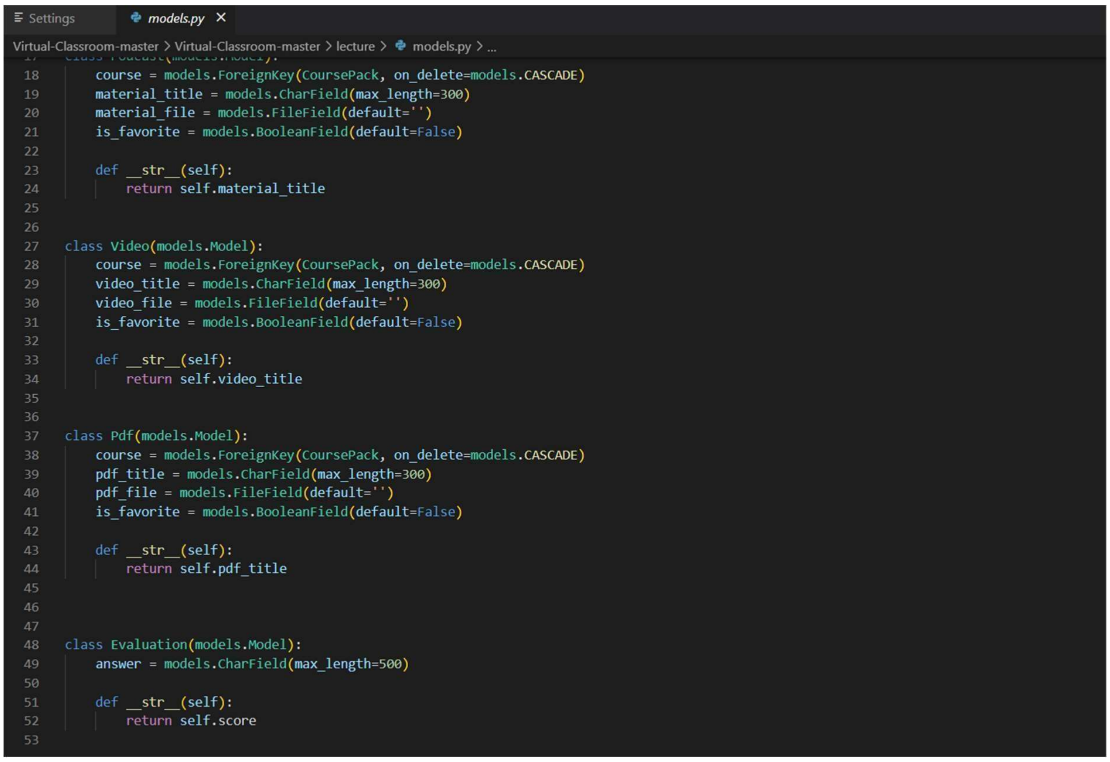
```python
from django.apps import AppConfig


class LectureConfig(AppConfig):
    name = 'lecture'
```


Settings
forms.py
Virtual-Classroom-master > Virtual-Classroom-master > lecture > forms.py > ...

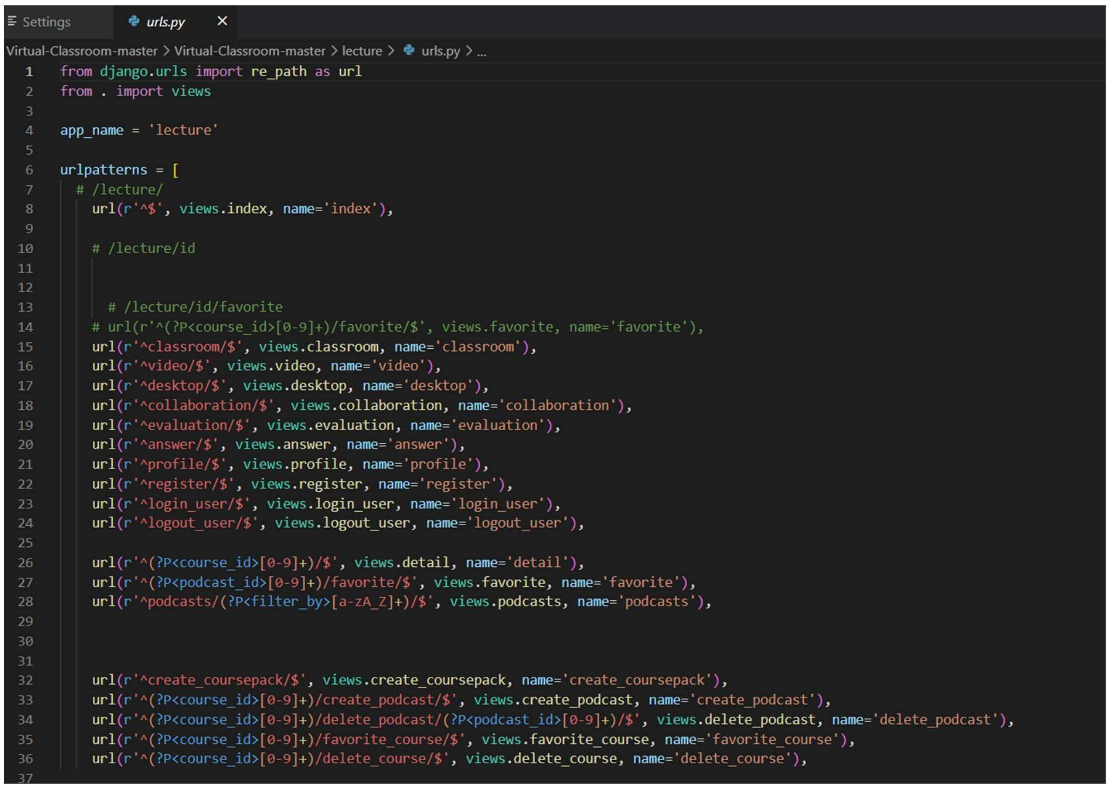
```python
from django import forms
from django.contrib.auth.models import User

from .models import CoursePack, Podcast, Video, Pdf, Evaluation


class CoursePackForm(forms.ModelForm):

    class Meta:
        model = CoursePack
        fields = ['instructor', 'course_title', 'course_code', 'thumbnail']



class PodcastForm(forms.ModelForm):

    class Meta:
        model = Podcast
        fields = ['material_title', 'material_file']


class UserForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ['username', 'email', 'password']


class EvaluationForm(forms.Form):
        post = forms.CharField()


```


Settings
models.py
Virtual-Classroom-master > Virtual-Classroom-master > lecture > models.py > ...

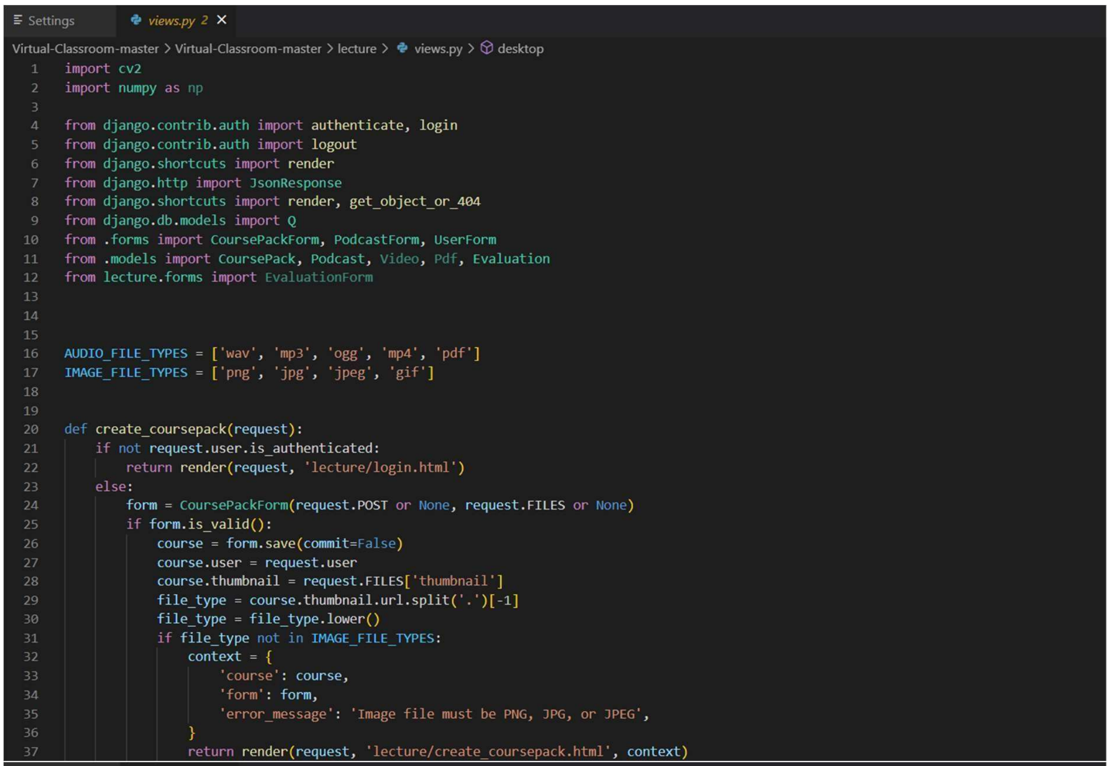
```python
from django.contrib.auth.models import Permission, User
from django.db import models

# Create your models here.

class CoursePack(models.Model):
    user = models.ForeignKey(User, default=1,on_delete=models.CASCADE)
    instructor = models.CharField(max_length=250)
    course_title = models.CharField(max_length=500)
    course_code = models.CharField(max_length=100)
    thumbnail = models.FileField()
    is_favorite = models.BooleanField(default=False)

    def __str__(self):
        return self.course_title + ' - ' + self.instructor

class Podcast(models.Model):
    course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
    material_title = models.CharField(max_length=300)
    material_file = models.FileField(default='')
    is_favorite = models.BooleanField(default=False)

    def __str__(self):
        return self.material_title


class Video(models.Model):
    course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
    video_title = models.CharField(max_length=300)
    video_file = models.FileField(default='')
    is_favorite = models.BooleanField(default=False)

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


Settings
models.py
Virtual-Classroom-master > Virtual-Classroom-master > lecture > models.py > ...

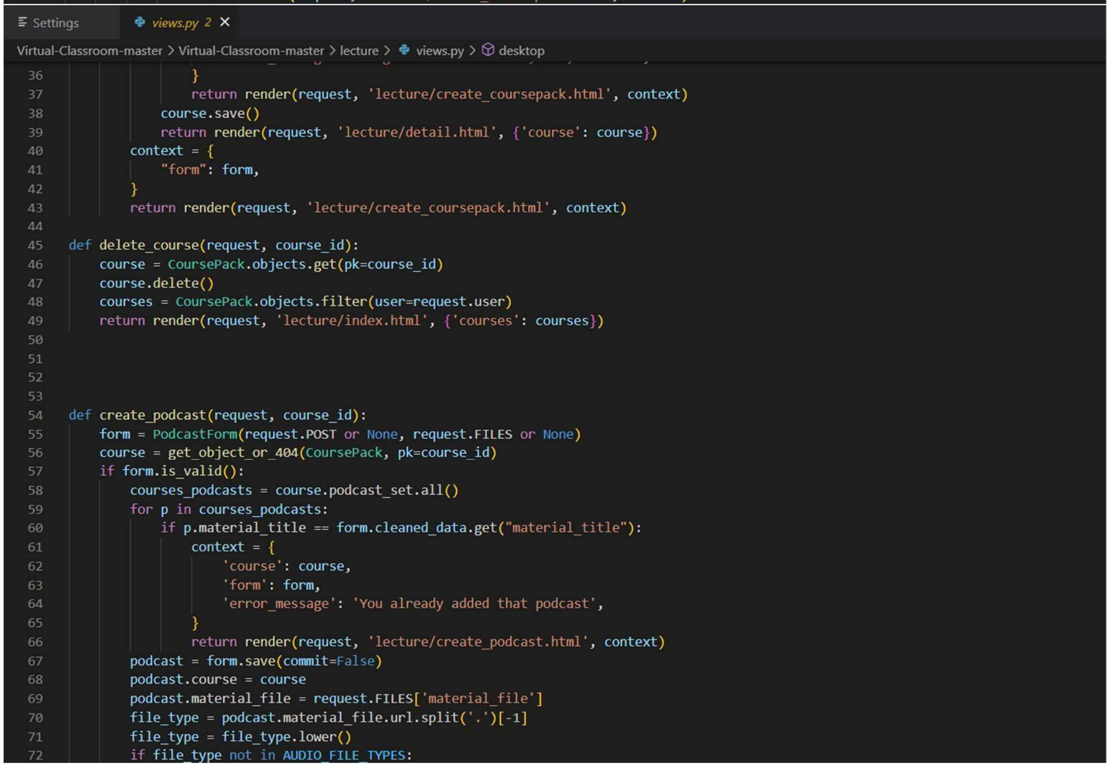
```python
    course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
    material_title = models.CharField(max_length=300)
    material_file = models.FileField(default='')
    is_favorite = models.BooleanField(default=False)

    def __str__(self):
        return self.material_title


class Video(models.Model):
    course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
    video_title = models.CharField(max_length=300)
    video_file = models.FileField(default='')
    is_favorite = models.BooleanField(default=False)

    def __str__(self):
        return self.video_title


class Pdf(models.Model):
    course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
    pdf_title = models.CharField(max_length=300)
    pdf_file = models.FileField(default='')
    is_favorite = models.BooleanField(default=False)

    def __str__(self):
        return self.pdf_title



class Evaluation(models.Model):
    answer = models.CharField(max_length=500)

    def __str__(self):
        return self.score

```


Settings
urls.py
Virtual-Classroom-master > Virtual-Classroom-master > lecture > urls.py > ...

```python
from django.urls import re_path as url
from . import views

app_name = 'lecture'

urlpatterns = [
  # /lecture/
    url(r'^$', views.index, name='index'),

    # /lecture/id


      # /lecture/id/favorite
    # url(r'^(?P<course_id>[0-9]+)/favorite/$', views.favorite, name='favorite'),
    url(r'^classroom/$', views.classroom, name='classroom'),
    url(r'^video/$', views.video, name='video'),
    url(r'^desktop/$', views.desktop, name='desktop'),
    url(r'^collaboration/$', views.collaboration, name='collaboration'),
    url(r'^evaluation/$', views.evaluation, name='evaluation'),
    url(r'^answer/$', views.answer, name='answer'),
    url(r'^profile/$', views.profile, name='profile'),
    url(r'^register/$', views.register, name='register'),
    url(r'^login_user/$', views.login_user, name='login_user'),
    url(r'^logout_user/$', views.logout_user, name='logout_user'),

    url(r'^(?P<course_id>[0-9]+)/$', views.detail, name='detail'),
    url(r'^(?P<podcast_id>[0-9]+)/favorite/$', views.favorite, name='favorite'),
    url(r'^podcasts/(?P<filter_by>[a-zA_Z]+)/$', views.podcasts, name='podcasts'),



    url(r'^create_coursepack/$', views.create_coursepack, name='create_coursepack'),
    url(r'^(?P<course_id>[0-9]+)/create_podcast/$', views.create_podcast, name='create_podcast'),
    url(r'^(?P<course_id>[0-9]+)/delete_podcast/(?P<podcast_id>[0-9]+)/$', views.delete_podcast, name='delete_podcast'),
    url(r'^(?P<course_id>[0-9]+)/favorite_course/$', views.favorite_course, name='favorite_course'),
    url(r'^(?P<course_id>[0-9]+)/delete_course/$', views.delete_course, name='delete_course'),
```

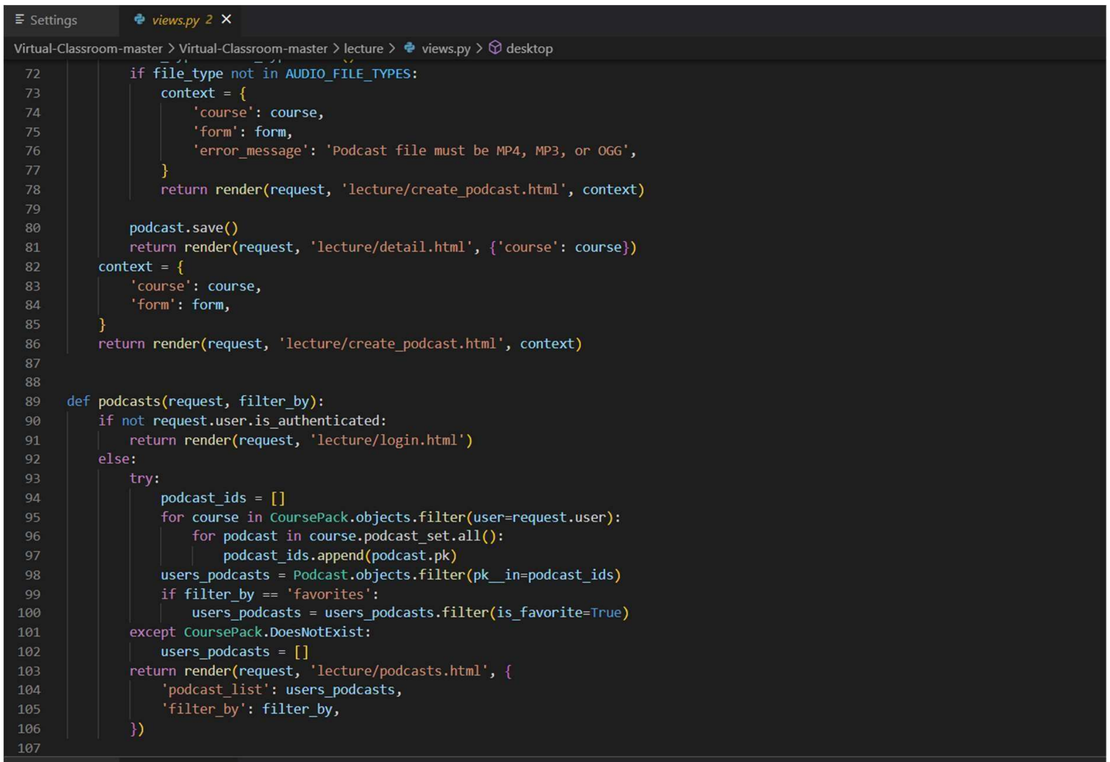

Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

```python
import cv2
import numpy as np

from django.contrib.auth import authenticate, login
from django.contrib.auth import logout
from django.shortcuts import render
from django.http import JsonResponse
from django.shortcuts import render, get_object_or_404
from django.db.models import Q
from .forms import CoursePackForm, PodcastForm, UserForm
from .models import CoursePack, Podcast, Video, Pdf, Evaluation
from lecture.forms import EvaluationForm



AUDIO_FILE_TYPES = ['wav', 'mp3', 'ogg', 'mp4', 'pdf']
IMAGE_FILE_TYPES = ['png', 'jpg', 'jpeg', 'gif']


def create_coursepack(request):
    if not request.user.is_authenticated:
        return render(request, 'lecture/login.html')
    else:
        form = CoursePackForm(request.POST or None, request.FILES or None)
        if form.is_valid():
            course = form.save(commit=False)
            course.user = request.user
            course.thumbnail = request.FILES['thumbnail']
            file_type = course.thumbnail.url.split('.')[-1]
            file_type = file_type.lower()
            if file_type not in IMAGE_FILE_TYPES:
                context = {
                    'course': course,
                    'form': form,
                    'error_message': 'Image file must be PNG, JPG, or JPEG',
                }
                return render(request, 'lecture/create_coursepack.html', context)
```

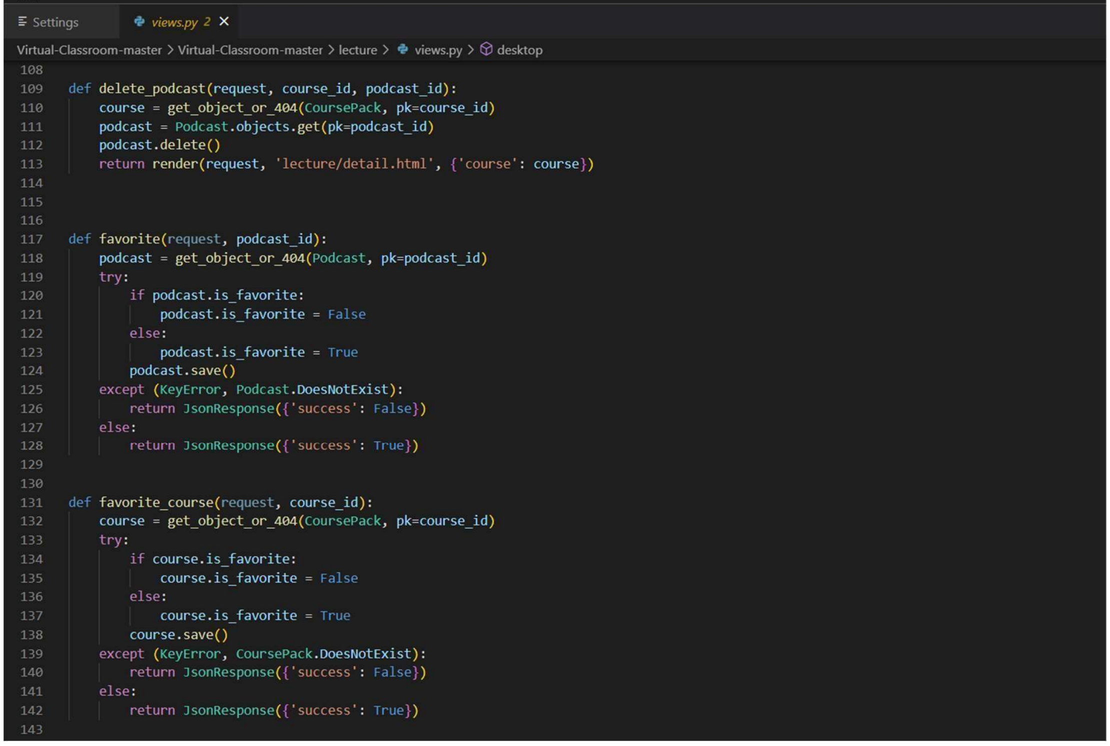

Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

```python
                }
                return render(request, 'lecture/create_coursepack.html', context)
```

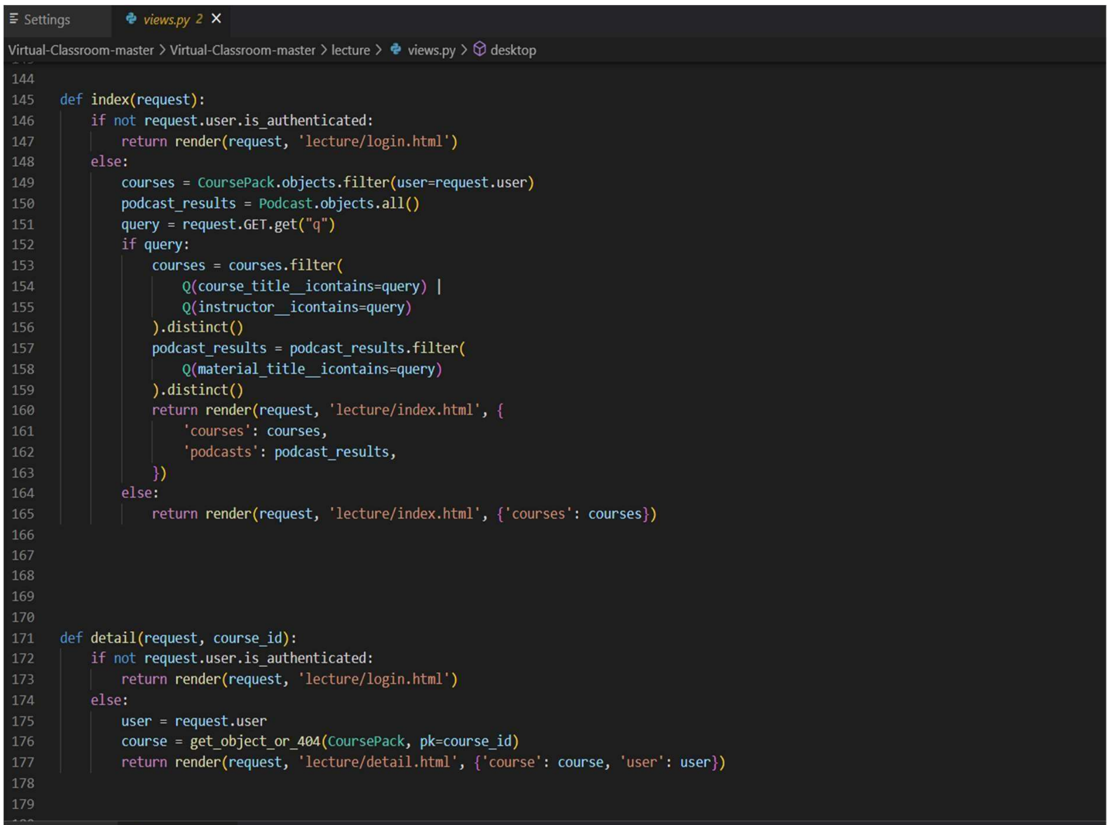
```python
            course.save()
            return render(request, 'lecture/detail.html', {'course': course})
        context = {
            "form": form,
        }
        return render(request, 'lecture/create_coursepack.html', context)

def delete_course(request, course_id):
    course = CoursePack.objects.get(pk=course_id)
    course.delete()
    courses = CoursePack.objects.filter(user=request.user)
    return render(request, 'lecture/index.html', {'courses': courses})




def create_podcast(request, course_id):
    form = PodcastForm(request.POST or None, request.FILES or None)
    course = get_object_or_404(CoursePack, pk=course_id)
    if form.is_valid():
        courses_podcasts = course.podcast_set.all()
        for p in courses_podcasts:
            if p.material_title == form.cleaned_data.get("material_title"):
                context = {
                    'course': course,
                    'form': form,
                    'error_message': 'You already added that podcast',
                }
                return render(request, 'lecture/create_podcast.html', context)
        podcast = form.save(commit=False)
        podcast.course = course
        podcast.material_file = request.FILES['material_file']
        file_type = podcast.material_file.url.split('.')[-1]
        file_type = file_type.lower()
        if file_type not in AUDIO_FILE_TYPES:
```


Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

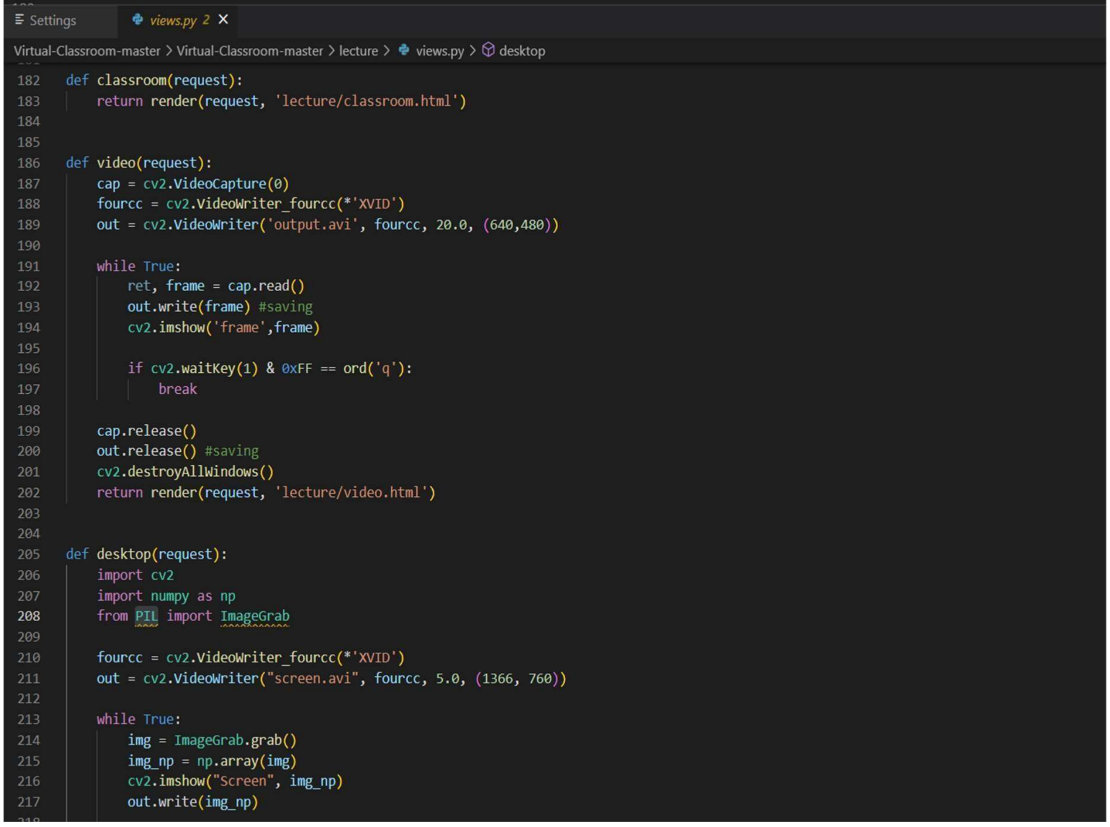
```python
        if file_type not in AUDIO_FILE_TYPES:
            context = {
                'course': course,
                'form': form,
                'error_message': 'Podcast file must be MP4, MP3, or OGG',
            }
            return render(request, 'lecture/create_podcast.html', context)

        podcast.save()
        return render(request, 'lecture/detail.html', {'course': course})
    context = {
        'course': course,
        'form': form,
    }
    return render(request, 'lecture/create_podcast.html', context)


def podcasts(request, filter_by):
    if not request.user.is_authenticated:
        return render(request, 'lecture/login.html')
    else:
        try:
            podcast_ids = []
            for course in CoursePack.objects.filter(user=request.user):
                for podcast in course.podcast_set.all():
                    podcast_ids.append(podcast.pk)
            users_podcasts = Podcast.objects.filter(pk__in=podcast_ids)
            if filter_by == 'favorites':
                users_podcasts = users_podcasts.filter(is_favorite=True)
        except CoursePack.DoesNotExist:
            users_podcasts = []
        return render(request, 'lecture/podcasts.html', {
            'podcast_list': users_podcasts,
            'filter_by': filter_by,
        })

```


Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

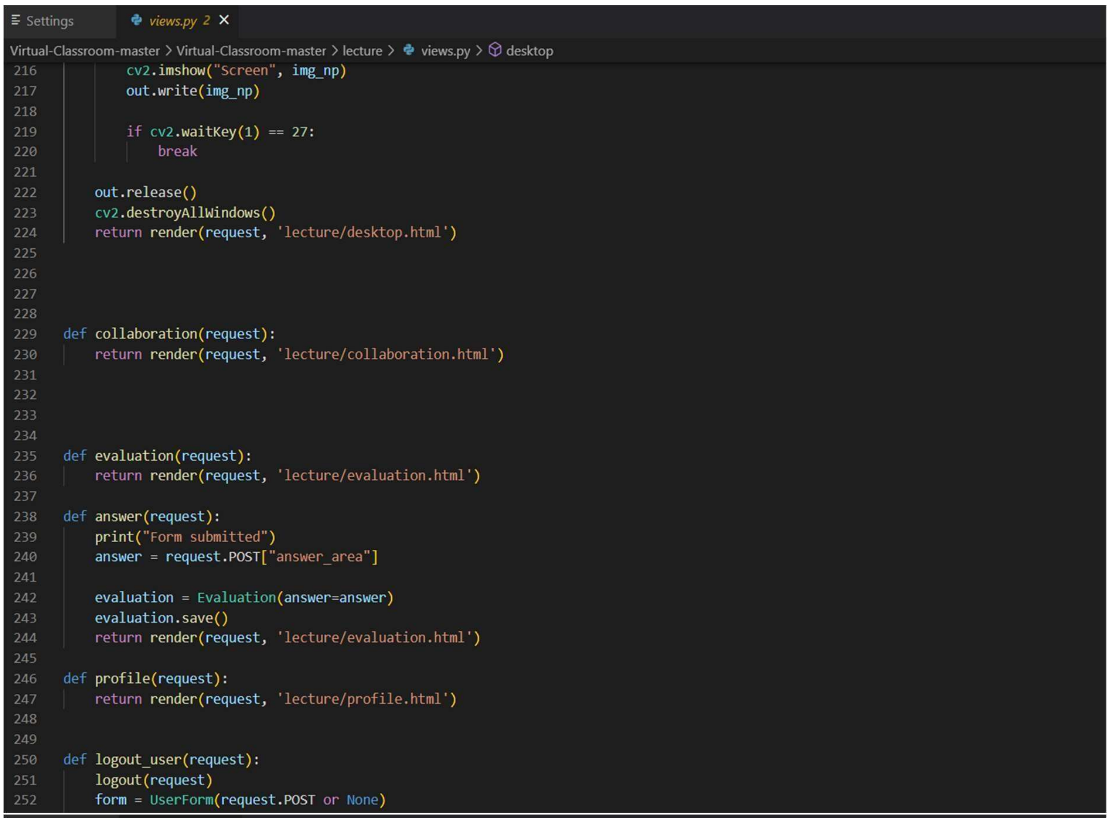
```python

def delete_podcast(request, course_id, podcast_id):
    course = get_object_or_404(CoursePack, pk=course_id)
    podcast = Podcast.objects.get(pk=podcast_id)
    podcast.delete()
    return render(request, 'lecture/detail.html', {'course': course})



def favorite(request, podcast_id):
    podcast = get_object_or_404(Podcast, pk=podcast_id)
    try:
        if podcast.is_favorite:
            podcast.is_favorite = False
        else:
            podcast.is_favorite = True
        podcast.save()
    except (KeyError, Podcast.DoesNotExist):
        return JsonResponse({'success': False})
    else:
        return JsonResponse({'success': True})


def favorite_course(request, course_id):
    course = get_object_or_404(CoursePack, pk=course_id)
    try:
        if course.is_favorite:
            course.is_favorite = False
        else:
            course.is_favorite = True
        course.save()
    except (KeyError, CoursePack.DoesNotExist):
        return JsonResponse({'success': False})
    else:
        return JsonResponse({'success': True})

```


Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

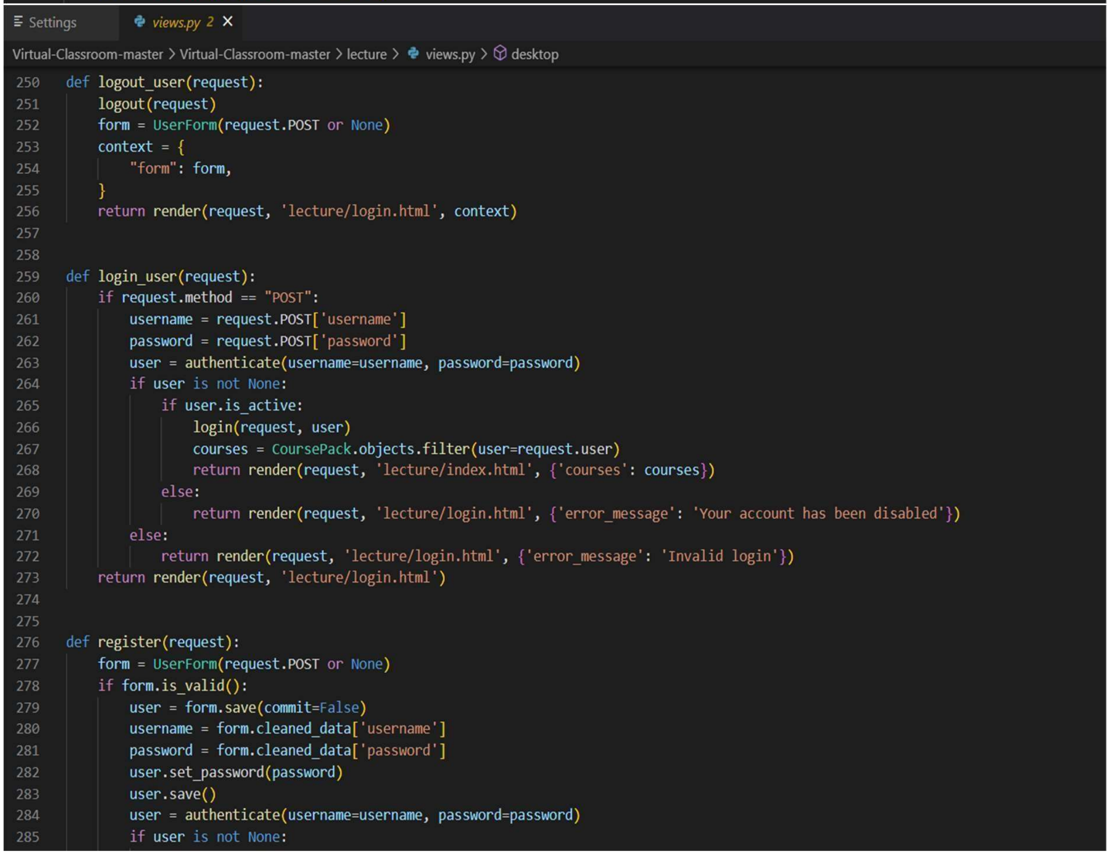
```python

def index(request):
    if not request.user.is_authenticated:
        return render(request, 'lecture/login.html')
    else:
        courses = CoursePack.objects.filter(user=request.user)
        podcast_results = Podcast.objects.all()
        query = request.GET.get("q")
        if query:
            courses = courses.filter(
                Q(course_title__icontains=query) |
                Q(instructor__icontains=query)
            ).distinct()
            podcast_results = podcast_results.filter(
                Q(material_title__icontains=query)
            ).distinct()
            return render(request, 'lecture/index.html', {
                'courses': courses,
                'podcasts': podcast_results,
            })
        else:
            return render(request, 'lecture/index.html', {'courses': courses})





def detail(request, course_id):
    if not request.user.is_authenticated:
        return render(request, 'lecture/login.html')
    else:
        user = request.user
        course = get_object_or_404(CoursePack, pk=course_id)
        return render(request, 'lecture/detail.html', {'course': course, 'user': user})


```


Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

```python
def classroom(request):
    return render(request, 'lecture/classroom.html')


def video(request):
    cap = cv2.VideoCapture(0)
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640,480))

    while True:
        ret, frame = cap.read()
        out.write(frame) #saving
        cv2.imshow('frame',frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    out.release() #saving
    cv2.destroyAllWindows()
    return render(request, 'lecture/video.html')


def desktop(request):
    import cv2
    import numpy as np
    from PIL import ImageGrab

    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter("screen.avi", fourcc, 5.0, (1366, 760))

    while True:
        img = ImageGrab.grab()
        img_np = np.array(img)
        cv2.imshow("Screen", img_np)
        out.write(img_np)
```


Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

```
            cv2.imshow("Screen", img_np)
            out.write(img_np)

            if cv2.waitKey(1) == 27:
                break

        out.release()
        cv2.destroyAllWindows()
        return render(request, 'lecture/desktop.html')




def collaboration(request):
    return render(request, 'lecture/collaboration.html')




def evaluation(request):
    return render(request, 'lecture/evaluation.html')

def answer(request):
    print("Form submitted")
    answer = request.POST["answer_area"]

    evaluation = Evaluation(answer=answer)
    evaluation.save()
    return render(request, 'lecture/evaluation.html')

def profile(request):
    return render(request, 'lecture/profile.html')


def logout_user(request):
    logout(request)
    form = UserForm(request.POST or None)
```


Settings
views.py 2
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop

```
def logout_user(request):
    logout(request)
    form = UserForm(request.POST or None)
    context = {
        "form": form,
    }
    return render(request, 'lecture/login.html', context)


def login_user(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                login(request, user)
                courses = CoursePack.objects.filter(user=request.user)
                return render(request, 'lecture/index.html', {'courses': courses})
            else:
                return render(request, 'lecture/login.html', {'error_message': 'Your account has been disabled'})
        else:
            return render(request, 'lecture/login.html', {'error_message': 'Invalid login'})
    return render(request, 'lecture/login.html')


def register(request):
    form = UserForm(request.POST or None)
    if form.is_valid():
        user = form.save(commit=False)
        username = form.cleaned_data['username']
        password = form.cleaned_data['password']
        user.set_password(password)
        user.save()
        user = authenticate(username=username, password=password)
        if user is not None:
```